AD-A059 849 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
OPTIMAL ROUTING WITHIN LARGE SCALE DISTRIBUTED COMPUTER-COMMUNI--ETC(U)
MAY 78 W H GREEN

UNCLASSIFIED AFIT-CI-78-69 NL

1 OF 3
AD A059849

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

AD A059849

DDC FILE COPY

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFIT-CI-78-69	2. GOVT ACCESSION NO. LEVEL	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: Optimal Routing Within Large Scale Distributed Computer-Communications Networks.	5. TYPE OF REPORT & PERIOD COVERED: Dissertation
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Major William Howard Greene	8. CONTRACT OR GRANT NUMBER(s): Doctoral Thesis
9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT Student at Texas A&M University, College Station, TX	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: AFIT/CI, WPAFB OH 45433	12. REPORT DATE: May 78; 1978
	13. NUMBER OF PAGES: 223p.; 210 Pages
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

DDC PROCURED SEP 11 1978 F

18. SUPPLEMENTARY NOTES

JOSEPH P. HIPPS, Major, USAF
Director of Information, AFIT

AUG 15 1978

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

012 200

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

78 08 21 012

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

78-69D

OPTIMAL ROUTING WITHIN LARGE SCALE DISTRIBUTED COMPUTER-COMMUNICATIONS NETWORKS

A Dissertation

by

WILLIAM HOWARD GREENE

Submitted to the Graduate College of
Texas A&M University
in partial fulfillment of the requirement for the degree of

DOCTOR OF PHILOSOPHY

May 1978

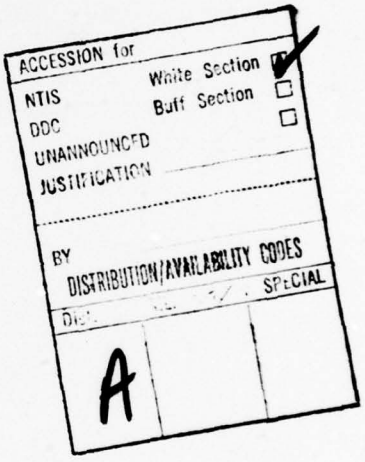
Major Subject: Computing Science

OPTIMAL ROUTING WITHIN LARGE SCALE DISTRIBUTED COMPUTER-COMMUNICATIONS NETWORKS

A Dissertation

by

WILLIAM HOWARD GREENE



Approved as to style and content by:

(Chairman of Committee)

(Head of Department)

(Member)

(Member)

(Member)

May 1978

ABSTRACT

Optimal Routing Within Large Scale Distributed
Computer-Communications Networks. (May 1978)
William Howard Greene, B.G.S., Omaha University,
M.C.S., Texas A&M University
Chairman of Advisory Committee: Dr. Udo Pooch

Much research has been devoted to developing efficient routing algorithms for data networks, particularly those referred to as packet switching, distributed networks. More recent developments have led to algorithms which are shown to produce optimum routes with non-looping characteristics. Several of these algorithms have been applied and are currently being used in operational networks. They are, however, restricted to use in small-to-medium scale networks because of excessive overhead which impacts both circuit bandwidth and nodal storage. This research investigates algorithms which function relatively independent of storage and bandwidth and are therefore adaptable to any size network.

The primary tool for demonstrating efficient algorithms lies with simulation. The importance of mathematical techniques, however, cannot be overlooked. Therefore, the initial phase of the research involves the investigation and development of abstract analytical concepts which provide an impetus to the design of the simulator. The approach employs a heuristic searching mechanism which requires that a network be described as a graph using the root-node-leaf notation. The level of the tree is equivalent to the known delay about a network at any particular node. The algorithm searches the tree down each leg, evaluating the path from each leaf to the destination node using → next page

heuristic information to determine the optimum path. This approach is combined with the classical decomposition-synthesis network evaluation technique to derive a formula for delay. Several heuristic measures applicable to this formula are evaluated by the simulator.

The simulation is described in fairly general detail. It is designed so that additional heuristics may be employed as they are developed. The GASP-IV simulation language was selected because it is FORTRAN based and because it allows a greater understanding of the network simulation process. Additionally, its output is thorough and easily interpreted.

The results of the simulator provide a foundation for sound conclusions about the operating characteristics of the evaluated algorithms on large distributed networks. The more important conclusions are the determination: 1) of an optimum search depth, 2) of an efficient heuristic measure of delay, and 3) that the best of the algorithms evaluated performs as good on all size networks as existing algorithms on small networks. These conclusions are based on measures of delay, queue lengths, and utilization of networks of various sizes. Though much research remains for large capacity networks of many nodes, statistical data generated by this research provide the basis for a relatively simple method for routing messages across large scale networks of the future.

ACKNOWLEDGEMENTS

I am deeply indebted to my chairman, Dr. Udo Pooch, for his advice and patience throughout my tenure at Texas A&M University. His uncanny, almost unique, ability to "see through the haze" resulted in many ideas at the inception and throughout the progress of this work. He has provided support, direction, and inspiration to complete tasks in a professional manner. Also, much gratitude is due to Dr. Glen Williams, who contributed a tremendous insight to the complications of simulating large networks, and to Drs. Rahul Chattergy and Gary Richardson for their support during the documentation phase.

This dissertation would be incomplete without acknowledging the technical support of Jerry Adkins and Terry Humphreys who, with their own research worries, found time to listen and assist on some of the more complex ideas. To Beth Baker, my sincere appreciation for her professional typing and drafting abilities and her dedication to see me through the final product.

Particular recognition is due to my wife, Francis, and children, Ann and Kirt, who helped and assumed added responsibilities in my stead, enabling me to devote my full time and effort to this research and dissertation.

To my wife, Francis, who

believes in the impossible.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

1. General

Computer networks have been classified as either a network of computers or a set of terminals connected to one or more computers (17). Most computer networks consist of hosts, terminals, nodes, and transmission links. A node generally refers to a computer whose primary function is to switch data. Computers used primarily for functions separate from that of switching data are referred to as hosts. Some designs permit the node and host functions to be performed by the same computer. Terminals are devices which interface the user to the computer or computer network and transmission links join this collection of subnet elements together to form a network (64). The transmission links and nodes along with the essential control software make up the communications subnet (35), usually referred to as the data network.

1.1 User Perspective

Prior to a review of literature on data communications networks, it is useful to have an understanding of:

(a) how designers, managers, and operators categorize data network users, and

(b) network design philosophy in meeting basic user requirements.

The Communications of the Association for Computing Machinery is used as a pattern for format and style.

1.1.1 User Characterization

1.1.1.1 Description of Factors (36)

A number of factors are related to the categorization of data communications users. The description of these factors has been limited to three broad categories which constitute a majority of the functions for which computer networks have an application.

1. Remote User Access: When users are not in the general vicinity of a computer system and need to have access to that system, they are considered to be remote users. Remote access is made available by attaching a terminal to some communications medium which is then interfaced to a computer.

Depending upon individual requirements, the user may interact with the computer for computational power or for access to data. The interactive user generally expects to receive a response within seconds of having generated some stimulus which would normally cause a response. However, the response to remote job entry, another user of remote terminals, is dependent on the time to process the entire job and the availability of the communications medium which is generally shared with other users.

2. Computer-to-Computer: Computers communicating directly with computers is creating the greatest demand for data communications media (5). Given the proper stimulus, a computer will generate bits of information for transfer at a much faster rate than the remote user described in the preceding paragraph. Portions of or even whole data bases can be conveniently transferred between computers without manual intervention. Computer-to-computer communication is the only factor requiring no direct manual interface.

3. Message Traffic: Much communication traffic is generated because some user desires to send a message to another user. The information transferred user-to-user fashion is referred to as message level traffic. A second type of message level traffic involves computer-to-user messages. Given a predetermined sequence of events, many computers are programmed to automatically transmit messages to their (remote) terminals. The computer-to-user communications are frequently classified as message traffic in the same context as terminal-to-terminal communications.

1.1.1.2 User Categories

Considering the above factors, description of users is logically divided into three categories. The first category shall be called the real-time user. A real-time computer system may be defined as one that receives and processes an input and returns a result with sufficient speed to affect the function at the terminal within an environmentally defined time frame (56). The user of a real-time computer system is a real-time user.

The second category shall be referred to as the teleconference user. These are users that desire to communicate a complete idea or concept as the communications entity. Messages of this nature are usually context sensitive and place no restrictions on the order in which symbols may appear in the message. As such, they are not easily adaptable for communicating with a computer. Therefore, user-to-user communications constitute the majority of message traffic; the users of these terminals are in a teleconference mode of communications.

The third and final category of user is referred to as the data-sharing user. The user in this context could be a computer (or the owner of the computer) which needs to share its data with other computers, or conversely, cause data to be shared with it. The data to be shared (communicated) could be as small as a few bits or as large as an entire data base. The data-sharing user will be a significant factor in the design of future computer networks.

1.1.2 Design Objectives

As with the design of any unit or system which depends on its users for operational funding, computer network designers must balance demand against capability (40). However, certain minimum requirements exist. In general, a computer network must be able to provide reliable, error-free communications within a reasonable time frame as defined by the user. The designer interprets these general objectives in the following terms:

a) Reliability (uninterruptable, error-free service)

b) Transparency (network operation should be invisible to the user)

c) Economy (minimum overhead and efficient use of media)

d) Convenience (user access methodology)

e) Security (as required by the user).

1.1.2.1 Reliability

Reliability is an inherent characteristic of any design effort. Reliability in a computer network refers to its ability to provide uninterruptable, error-free service. Uninterruptable service is greatly dependent upon a design philosophy which addresses the question, "To what

extent should alternate transmission paths and backup equipment be provided?" The answer to this question generally requires a statistical analysis of cost versus hardware reliabilities and a queuing analysis of load factors generated by potential users. Load factor analysis has played a major role in the design of new computer switching networks discussed later.

1.1.2.2 Transparency

Transparency is one of the most important design features of any communications network. Because of technological constraints, most networks in existence do not possess total transparency; that is, there are certain values (bit configurations) which cannot appear in the text of a message because of automatic hardware control actions which will take place. For that purpose, the American National Standards Institute (ANSI) has set aside, as a standard, certain bit values to be used exclusively for hardware control. Since they are values for which no other use has been designated, one could conclude that the transparency problem has been circumvented. Such is not the case, however, since certain types of user schemes, i.e., facsimile, graphics, and raw satellite weather and photographic data will generate fields of unknown values with a high probability that designated control values will be contained somewhere in the text. Transmitting this type of information over computer networks which are not totally transparent will cause unpredictable and probably unsuccessful results.

Technology has only recently advanced so that total transparency is a realistic design goal. The problem remains, however, to have a scheme standardized so that user computers of many types may communicate without

major interface redesign. An ANSI group has been established for that purpose and a standard has been proposed (78) for review by the appropriate committee members.

1.1.2.3 Economy

Expediency of communications is an ever increasing demand. Economic considerations have seriously constrained any significant widescale improvements in data transmission speeds in the near future. The problem arises because the primary communications medium, landlines, was designed for voice communications at bandwidths less than that required for wideband data transmission. The upgrade of existing landline networks or the installation of a new medium capable of communicating wideband data would involve astronomically high costs to be responsive to the current demand rate of change. Except for backbone circuits, economic considerations will inhibit the widespread use of the high speed data rates through the life span of the existing landline systems.

A computer network should function with minimum overhead. Since, in general, computer power exceeds that necessary to keep links saturated, overhead is essentially a discussion of link loads. With less overhead on a link, more efficient utilization can be made of available computer power with a corresponding increase in effective transmission rate. What is link overhead? It is that portion of a message exclusive of text required to communicate between computers. A certain amount of control information must be attached to each message for the receiving computer to interrogate in order to determine the proper course of action for the message. The efficient design of a scheme for transmitting control information will tend to minimize overhead without restricting flexibility.

This design problem is compounded by differing media and computer interface characteristics. The complexity of accounting for these divergent factors makes it no simple task to design a "best" scheme for minimizing overhead.

1.1.2.4 Convenience

Though not so readily apparent, a critical design factor is user convenience. If accessing a specific type of computer network is troublesome, that network is competitively placed at a disadvantage to other types of networks. The reputation of a network, regardless of its other features, can be quickly tainted through the only physical interface to the user. The designer must attempt to keep access procedures simple in order to retain customers.

1.1.2.5 Security

Data transferred through a computer network should be securable if so desired by the user. Current computer technology constrains the amount of security the user can expect from the communications system. Since there are no perfectly secure computer systems, the network design should consider various computer design features which tend to enhance security. Some computers are more securable than others and should be appropriately weighted for that purpose. A combination of sophisticated hardware and simple software increases the chance of providing a secure data communications media.

1.1.2.6 Other Considerations

The management of a geographically dispersed system which potentially interfaces with thousands of users possessing equipment from hundreds of different vendors each with its own interface characteristics, poses unusual problems. Under current guidelines it is improbable that any one

organization could collect the necessary talent to cope with engineering and "finger-pointing" problems that are inevitable in a multivendor environment without pricing the service out of reach. In reviewing the literature one gains an intuitive feeling that a common technological base is necessary between users and vendors for the independent success of a computer network system. Such a base, in the form of internationally established computer interface standards, is slowly evolving through a group of user, government, and vendor representatives called the International Standards Organization (ISO) (3). The required standards are expected to be published within the next two year time frame. Though other management problems must be overcome, none possess the novelty or approach the complexity of that described above.

There are certain social implications of linking data bases which contain information of one form or another on every U.S. citizen. Regulatory bodies have already indicated their awareness of the problem by invoking a privacy act which restricts release of private personal information by federal agencies (77). As a result, security is now being given increased emphasis in nongovernment systems (security has always been a primary design factor in government systems). As previously stated, security has certain technological constraints which, at least for the near future, will inhibit the use of general computer networks for extremely sensitive communications.

1.2 Research Objective and Plan

This dissertation is concerned with the development of data communications network routing algorithms which are independent of network size. The algorithms should possess many of the characteristics of previously

developed algorithms which allow dynamic rerouting of messages for varying system loads and minimum time for network traversal. But, contrary to the previous developments which possess these characteristics, procedures developed herein should not require the large storage arrays for determining the minimum time path to other nodes or subnets in the network nor should they require the overhead for preventing looping of messages within the network. The purpose of this research, then, is to develop a routing algorithm which can be used for any size network while retaining as many of the desirable properties of existing dynamic routing algorithms as possible.

In Chapter II a survey of the literature related to the routing problem is provided. Detailed concepts and terminology are presented as support material to succeeding chapters.

Chapter III is used to describe a broadly defined analytical model of investigated algorithms. Many aspects of the analytical model necessarily remain abstract to allow flexibility for investigating the various alternatives encountered. Too, the analytical model is incomplete since mathematical techniques of sufficient power to completely describe such networks have not yet been developed. As a result of mathematical deficiencies much development has occurred through simulation. A description of the simulator developed in support of the research is presented in Chapter IV.

An algorithm which is adaptable to any size network is examined in Chapter V. A detailed analysis is provided through the use of data obtained from the simulation phase.

Finally, a summary of the results of the research, conclusions, and proposed recommendations and suggestions for further research are presented in Chapter VI. Applicable queuing theory, simulation data, and a program listing of the simulator model are contained in Appendices A, B, and C, respectively.

CHAPTER II

LITERATURE SURVEY

2. Overview

The complexity of computer networks has taken a dramatic upswing along with the more significant developments in electronic technology such as medium and large scale integrated circuitry and microprocessors (53). Along with this upswing in complexity, there have evolved several sophisticated methods of classifying networks based upon message routing schemes, topology, network functional aspects, and combinations of these. A detailed discussion of these classifications is provided after a brief review of history on data networks.

2.1 Historical Background

Initial computer installations, in the late 1940's and early 1950's, were either dedicated to a particular research problem or used for finance accounting. Since that time, the explosive growth of computer technology, both hardware and software, has placed the computer into a wide spectrum of applications. Individual installations, however, were prone to develop sophisticated software not readily transportable to the wide diversity of machines and interfacing software. As a result, users frequently were required to develop duplicate software packages which could be economically adapted to their own installation.

During the same time frame, military planners were becoming increasingly concerned with providing survivable, low-delay communications to support advanced weapon systems. High speed digital computers were employed to meet the critical response times for air defense communications.

A computerized air defense system called SAGE (72), installed in 1958, was the first attempt to interconnect computers on a large scale. Later, in the early 1960's, the Automated Digital Information Network (AUTODIN) was installed to provide the Department of Defense with quasi-survivable data communications (4). Development of AUTODIN provided the initial impetus toward the design of message routing schemes (60,70).

The experience of the military planners and supporting contractors led to development of the early systems relying heavily on common carriers rather than dedicated lines (CYBERNET (54) and DATRAN (8,26,34)). Store and forward networks first began to appear in 1969 as a cost effective, common carrier approach in response to the popularity of the many time-sharing systems on the market at that time (34). Though time-sharing systems allowed many users to simultaneously interact with a machine, usage was not uniform throughout the day (51,57).

The nature of scheduling one's activities led to relatively short periods of peak usage during which machines became strained and even saturated. Administrative procedures for scheduling terminal usage produced better utilization of the machine during slack periods but did not prevent periodic peaks from saturating the system. The advancement of network technology was encouraged as a solution to the problem of program transportability and peak machine loadings.

In 1967, Roberts (71) proposed a method for better utilization of resources through load-leveling, elimination of functional duplication, and specialization of hardware and software. The Advanced Research Projects Agency (ARPA), largely as a result of Roberts' research, proposed a concept building a distributed switchpoint system having non-homogenous hosts and lines. Such a network, according to ARPA, would have

specialized capabilities throughout the system to meet unique requirements of each user. The research efforts through ARPA have been the impetus for the rapid growth of computer networks during the 1970's.

2.2 Functional Classifications

There are basically three technologies for interconnecting computers, each with its own distinguishing characteristics. "These technologies are distinguished by the manner in which resources are allocated in support of communications ..." (43) and therefore take on the view of the designer who must make maximum utilization of the resources. The technologies are:

a) circuit switching,

b) message switching, and

c) packet switching.

A conceptual description of each is contained in the following paragraphs along with some comparisons and applications. The reader desiring more detail is referred to Kimbleton and Schneider (43) and the many references contained therein.

2.2.1 Circuit Switching (15,16,21,28,38,43,57)

Circuit switching is analogous to the telephone (voice) switching networks where a complete circuit or route is established prior to the start of communication by the users (12). It comes in two forms, manual or automatic, both involving the exclusive dedicated use of circuits. The manual switching of circuits is used mostly with remote terminals, generally for interactive type communications. In this mode, the user dials the proper telephone circuits for access to the desired computer system. If the path is unacceptable or if access to another computer

is desired, the user terminates the existing connection and redials (switches) to another circuit. Automatic circuit switching systems, on the other hand, require the use of electronic switching mechanisms which automatically connect the required circuit when pulsed with the proper sequence of bits. Both modes of circuit switching experience line contention delay when distant-end user circuits are busy.

Though widely used for individual remote terminal access, circuit switching has not been considered as having significant network potential, both in terms of efficiency and economic practicality. However, the not-to-distant future holds some evolutionary steps in switching speeds of solid state devices which will justify a reevaluation of circuit switching as a viable alternative in network design (43). Networks which use some form of circuit switching are DATRAN (57) and TYMNET (76).

2.2.2 Message Switching (12,21,38,45,63,67)

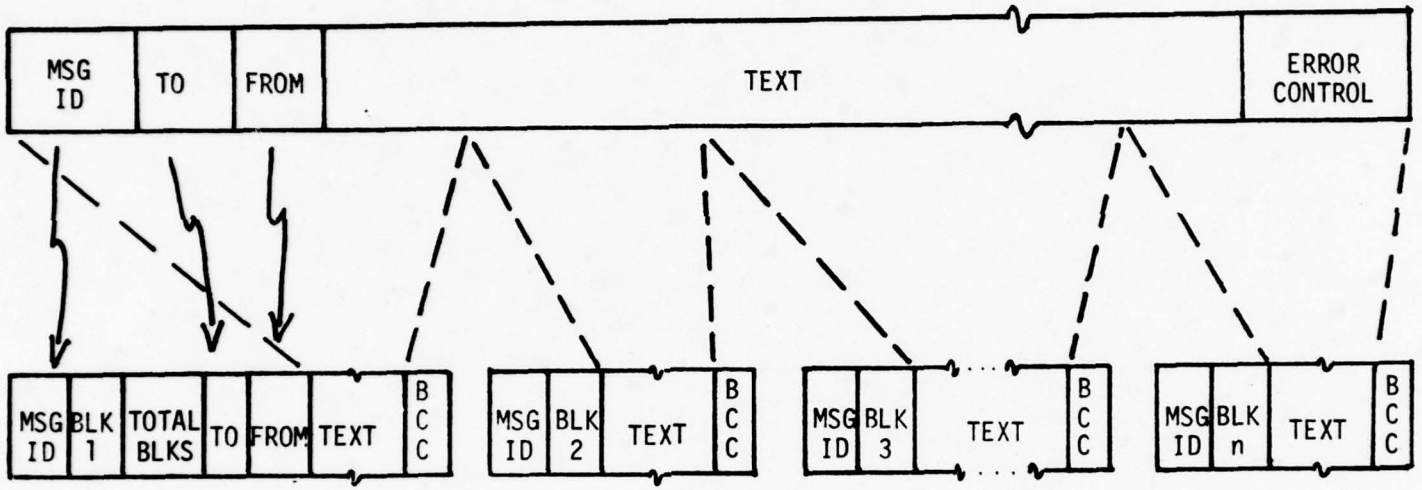
A message is defined to be a logical unit of information from the viewpoint of the user (43). Thus, telegrams, programs, and data files are examples of messages. A message switching subnet may be regarded as a collection of physical circuits interconnected by computer switches which have the ability to interrogate message control fields for determining subsequent action. It differs from circuit switching in that circuits are no longer dedicated for exclusive use. Message switching is, therefore, considered to be less expensive since circuit costs are amortized among the users sharing the system. However, any significant increase in switching speeds of evolving technology could cause circuit switching to have a cost advantage. AUTODIN is a prime example of a message switching network (4).

Messages within a message switching network are communicated between switches on a message-by-message basis. This means that a message, broken into blocks of data, must have been either transmitted and received in its entirety or canceled across a link before the next message can be transmitted. Each block of the message must be transmitted in its proper sequence so the receiving switch can rebuild the message and verify to the sending switch that it has taken accountability for the message (Figure 1). (Note that only the initial block has sufficient control information for further routing.) If the message is not accepted, it must be retransmitted until accountability can be verified by the receiving switch. Direct access storage devices (DASD) are employed to prevent unreasonable restrictions on message lengths and to store messages in cases of circuit failure or heavy loading. Because of these characteristics, message switching is frequently referred to as store-and-forward message communications. Generally, real-time statistics are not kept and, thus, dynamic load balancing is made impractical. For that reason, heavy loading on any one specific circuit is not normally sufficient criteria to seek alternate traffic routing. Contrary to the operation of most packet switching networks (described below), alternate routing in message switching networks is reserved almost entirely for circuit or switch failure.

2.2.3 Packet Switching (25,28,39,43,58)

A new technique for data communications which has evolved over the past ten years is called packet switching. As with message switching, each message is subdivided into blocks of data, called packets, for circuit transition. However, in packet switching each packet has attached

Figure 1. Example format for message switching network.

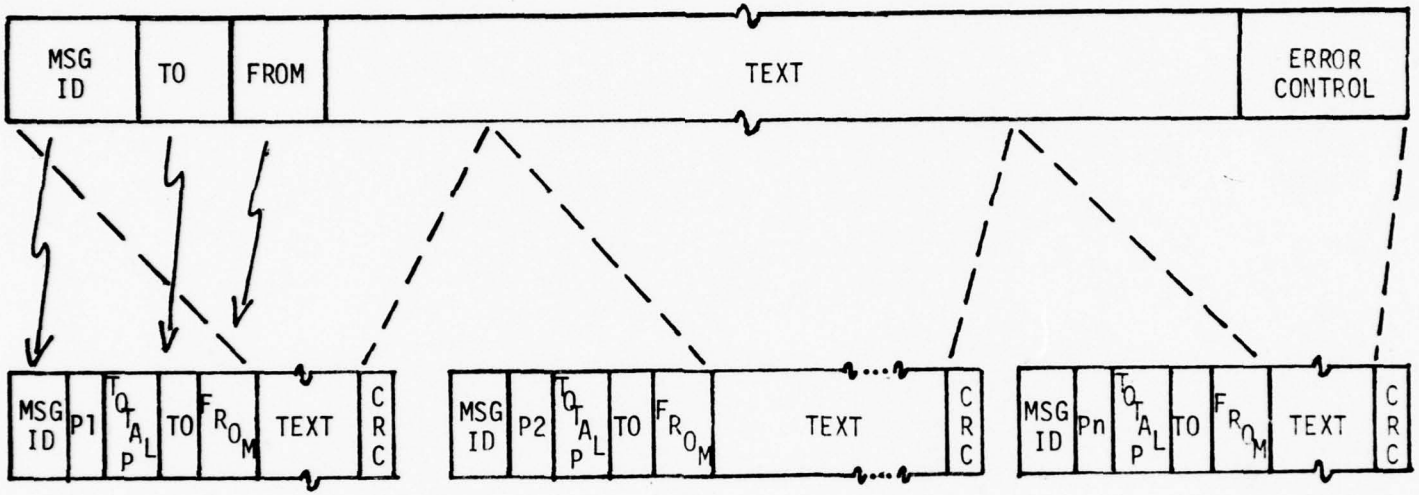


BCC - Block Check Character

to it sufficient control information that the packet can be communicated across a network independent of other packets belonging to the same message (Figure 2). The following example, whose notation can be traced to Kleinrock (50), describes how a message might traverse a packet switching network.

Consider a five packet message (Figure 3) which must traverse the six switch (node) network from A to F. The path of each packet can be represented by a set of ordered pairs to designate the circuit between two nodes, i.e., the circuit, P, from A to C is designated P_n = (ac) and the circuit from C to D is designated P_n = (cd) such that the corresponding path from A to D becomes P_n = (ac,cd), where n is a unique number assigned to a particular packet traversing path P. Note that all links communicate in both directions such that a packet going from C to A would have path P_n = (ca). Through a complex mechanism of evaluating individual circuit and switch loads, packet switching computers are generally programmed to dynamically alter routing of packets to the circuit of minimum loading. For simplicity, assume that unique numbers 1 through 5 have been assigned to packets one through five, respectively. The originating switch, node A, may determine that packets should be transmitted alternately to nodes B and C such that P_1 goes to B and P_2 goes to C, etc. Now node B may determine that P_1 and P_5 should go to D and P_3 should go to E. Node C, on the other hand, has also sent P_2 and P_4 to D. Node D, now with four of the five packets, dynamically determines that it can relieve its load by alternately transmitting the packets to nodes E and F. The end result is that packets arrive at F out of sequence. This presents no problem, however, since node F has been programmed to hold final delivery of the message until the entire message is received. If node F or any

Figure 2. Example format for packet switching network.



P - Packet
C - Cyclic redundancy check

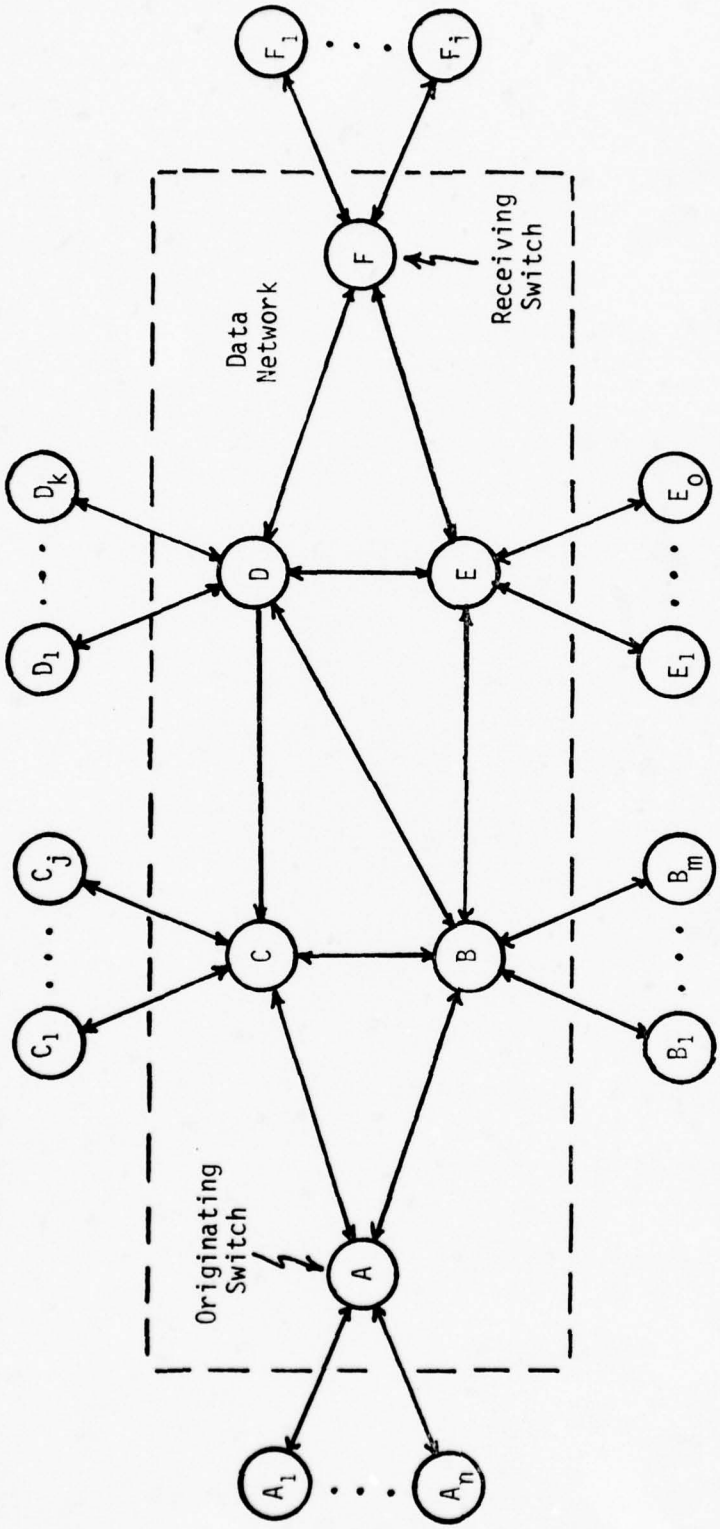


Figure 3. Example of packet switching network.

node between A and F received a garbled packet, the sending node will be requested to retransmit the packet. However, if F has not received all of the packets after a specified time interval, node A will be requested to retransmit the missing packets having assumed that the first transmission was lost in noise, equipment failure, etc. The following path descriptors for packets in the example emphasize the flexibility with which a message may traverse a packet switching network:

P_1 = (ab,bd,de,ef),

P_2 = (ac,cd,df),

P_3 = (ab,be,ef),

P_4 = (ac,cd,de,ef),

P_5 = (ab,bd,df).

The ability to immediately transmit a packet without waiting for a complete message and the ability to dynamically adjust to varying load factors minimize resource requirements at each node (58). Nodes which become saturated, however, are generally programmed to reject traffic until their traffic load is normalized. The overall effect is a decrease in system cost with a corresponding effect on user rates. Example networks using packet switching technology are ARPANET (50,72) and MERIT (5,75).

2.3 Topological Classifications

Network topology, as a means for categorizing data communications networks, evolved from graph theory. A detailed introduction to graph theory and its relationship to computer science is provided by Deo (22). Topology refers to properties of a network which are independent of its size and shape, such as the connection pattern of links and nodes (21).

Graph is the mathematical term for network, i.e., a collection of points joined by links. Terms borrowed from graph theory include node, which is analogous to the network computer switch; link, which provides a connection between any two nodes and is analogous to the communication channel; and path, which represents the physical media for communications of intelligence across the network. This terminology forms the basis for extensive research into the various topological classifications of data networks, i.e., centralized, decentralized, and distributed (Figure 4). The following paragraphs provide the distinguishing characteristics of each of these classifications.

2.3.1 Centralized Networks

The centralized network is essentially a star topology, and is the simplest of arrangements where switching has been introduced into the network (38). This topological scheme requires that a link be dedicated for communications between the central node and each terminal during any period of operation for a terminal.

The reliability of the centralized network is highly dependent upon the central switch (node). Its failure suspends all activity in the network whereas individual link failures affect only a single device per link. Any significant increase in reliability requires duplication of the switching function, a relatively high expense endeavor.

Geographically dispersed terminals frequently lead to the use of concentrators or multiplexors. Indeed, it is very unlikely that all terminals would be simultaneously active in a system of terminals joined to a central computer. A hardware switch may be justified where a subset of the terminals remotely connected to a central computer are located in the same geographical vicinity (Figure 5). The objective is to obtain

Figure 4. (a) Centralized network, (b) Decentralized network, (c) Distributed network.

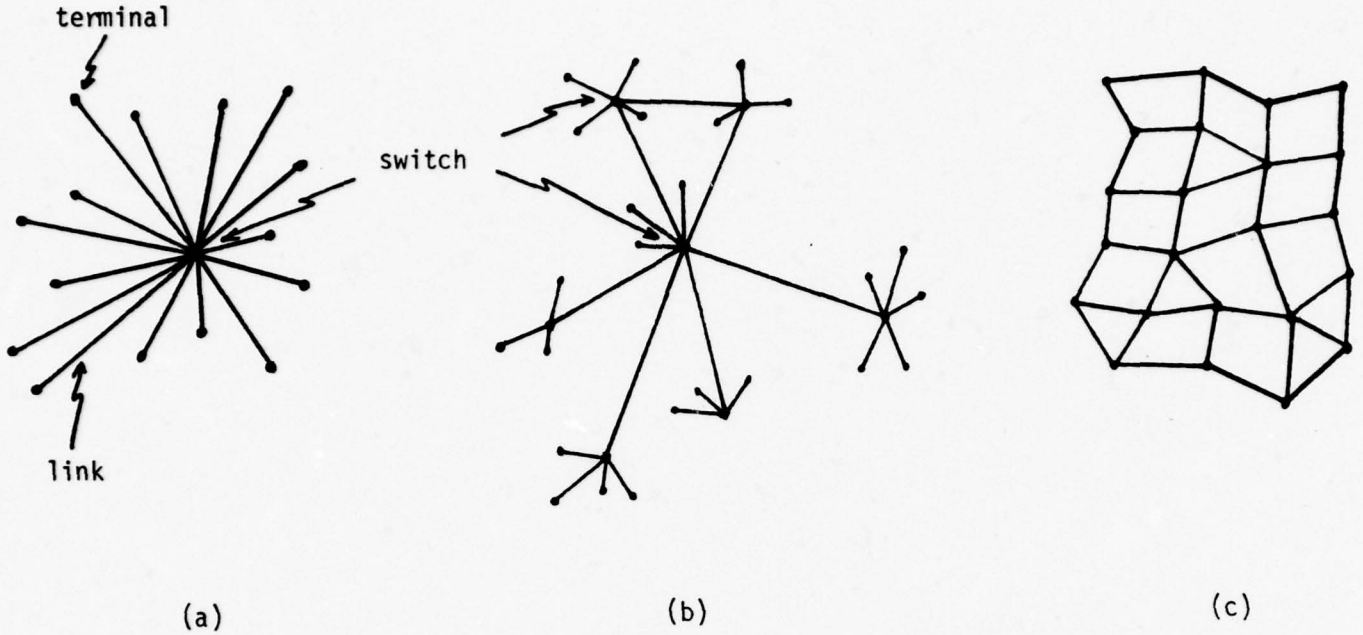
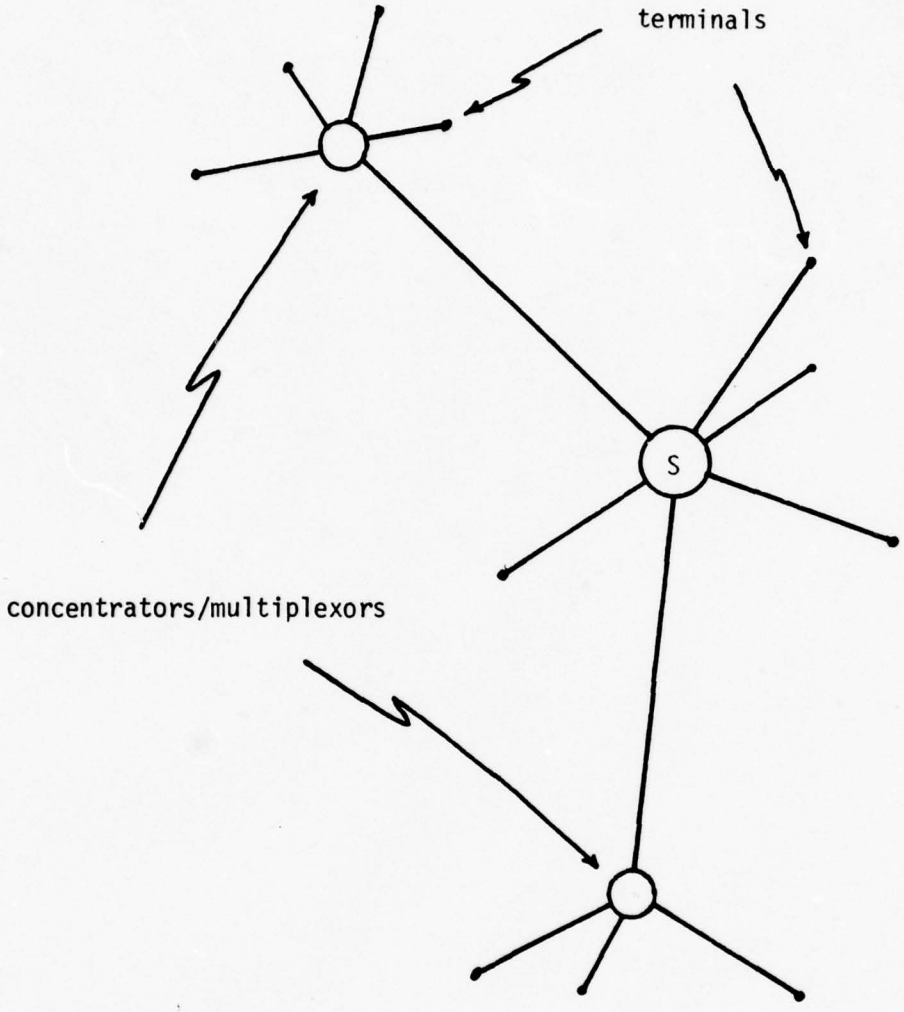


Figure 5. Centralized network with concentrators/multiplexors.



more efficient link utilization at the expense of an occasional delay in turn-around time. The switch is referred to as a multiplexor where the information transfer rate required never exceeds that formally demanded by one link (21). Where the potential input capacity exceeds link capacity, the switch is called a concentrator. The latter of these requires some storage capacity to compensate for occasions where instantaneous input rates exceed link capacity. Concentrators are also used to merge several low speed links into one high speed link. The switching power of concentrators and multiplexors is totally dependent upon an operational central node.

2.3.2 Decentralized Networks

The distinction between centralized and decentralized networks lies in the organization of the switching function (Figure 4). Decentralized networks may be viewed as an expanded centralized network possessing multiplexor/concentrators whose switching power is, to some extent, independent of that of any other node. Graph theorists would refer to such a network as a mixture of star and "mesh" components, where a mesh is a completely enclosed region (22) (note that Figure 4b contains a mesh component).

The added reliability of decentralized switching power comes at the expense of additional computers (nodes) and corresponding connecting links. This added reliability generally comes in some (though not elaborate) form of alternate routing, that is to say, not every path will be duplicated (21). Within reason, the reliability of networks can be made as high as desired, however, by duplicating paths with the addition of corresponding links and nodes. The existence of at least two disjoint paths between every pair of nodes describes what Baran (6) refers to as

the distributed network (Figure 4c), discussed in some detail in the following paragraphs. It should be noted at this point that recent papers (32,35,50,59) have dropped any reference to the decentralized network, combining its description with that of distributed networks.

2.3.3 Distributed Networks

Much has been written about distributed networks since networking of computers was first envisioned in the early 50's starting with telephone systems. Practically every paper published since that time about computer networks has made some reference to the distributed class of networks. The evolution of this intriguing and complex maze of computers has led to consideration for networks with thousands and even tens of thousands of nodes (39,50). Therein lies much of the on-going research on networks today.

The distributed network, as originally envisioned, consists of a set of mesh subnetworks where each node is connected to a minimum of three other nodes. Each node possesses the capability to systematically switch between connected links according to some prescribed routing algorithm (several of which are described in section 2.4) which maximizes the capacity of a particular network. Distributed networks evolved from attempts to define military communications systems suitable for operating in hostile environments (6,9). Because of the inherent reliability for continuity of operation, the distributed network, using packet switching techniques, is generally considered to have the greatest potential for networks of the future. Much of the ensuing discussion in the following paragraphs is attributed to recent work by Boorstyn and Frank presented at the First Joint IEEE-USSR Workshop on Information Theory, Moscow, USSR, December, 1975 (11).

Performance of distributed data networks is characterized by parameters of cost, throughput, response time, and reliability. The design of the network should consider properties of its nodes along with the network's topological structure. Important considerations are (29):

a) Node Characteristics
    1) Message handling and buffering
    2) Error control
    3) Flow control
    4) Routing
    5) Node throughput
    6) Node reliability

b) Topological Characteristics
    1) Link location
    2) Link capacity
    3) Network response time
    4) Network throughput
    5) Network reliability

For small or medium sized networks (up to 20 or 30 nodes), the typical structure is fairly homogeneous with identical hardware and software at each node. Because of high packet overhead and storage for delay vectors in using global routing procedures, larger networks require the use of alternatives such as topologies with embedded hierarchies. For a two-level hierarchy, the highest level can be thought of as the "backbone" network and the lower hierarchy as a set of subnets which access the backbone network through one or more high-level nodes that act as gateways. Boorstyn et.al. has classified four general problems which must be considered for multilevel networks:

a) Preliminary clustering of user locations,

b) Selection of nodal processor locations,

c) Backbone topological design for the upper levels,

d) Local access design.

It is item c, backbone topological design, which currently poses the greatest challenge for designers of large-scale distributed networks and which is currently receiving the most attention in data network research.

2.4 Routing Classifications

Extensive research has been done on the design and modeling of network routing algorithms by Prosser (69,70), Boehm and Mobley (9), Doll (23), Gerla (35), Metcalfe (62), McQuillan (59), Kleinrock (50), and many others. The descriptions contained herein are attributed primarily to the doctoral work of Fultz (32) who has consolidated an excellent taxonomy for classifying network routing algorithms (Table 1). As will be noted, there is considerable overlap between classifying networks according to the type of routing algorithms used and according to the nature of their topology.

Though not specifically identified, several of the algorithms discussed in this section are extensions of routing mechanisms developed for voice (circuit) switching networks as they evolved in the 1950's (6). An extensive literature review indicates that since that time little has been done to improve on circuit switching algorithms for data communications purposes. Therefore, the ensuing discussions are devoted to a review of routing schemes for packet/message switching networks.

2.4.1 Deterministic Algorithms

Deterministic routing algorithms derive routes according to some given deterministic decision rule which produces loop-free routing, i.e.,

Table I. Classification of routing algorithms (32)

1. Deterministic Flooding — All Flooding — Selective Fixed Split Traffic Ideal Observer
2. Stochastic Random Isolated — Local Delay Estimate Isolated — Shortest Queue + Bias Distributed — Periodic Update Distributed — Asynchronous Update
3. Flow Control Isarithmic Buffer Storage Allocation Special Route Assignment

messages (packets) can never become trapped in closed paths. This type routing policy consists of four basic techniques, some of which are subdivided into more descriptive categories.

2.4.1.1 Flooding Techniques

Flooding appears to be the simplest of all algorithms (9,21,45). It requires neither large storage for maintaining current routing information nor statistical data for building sophisticated delay measuring mechanisms, etc., required by stochastic and flow control techniques. A node must only remember the link a message arrived over. It then retransmits the message over all connected links except the link from which the message was received. After having circulated through the network for a prespecified period, a message is then returned to the originator as confirmation that the flooding cycle has been completed for that message. *In terms of minimum* delay, flooding always produces an optimum routing policy as the network is placed in operation. After an initial stabilization period, it quickly succumbs to congestion because of the excess traffic in the network. As well as having been proposed for various military applications (23), it has been suggested that flooding might be used as an initial "path finder" to derive routing and path delay statistics required for other techniques (21). Efficiency considerations, in general, though, rule out the use of flooding techniques as a day-to-day policy for network routing (9).

2.4.1.2 Fixed Techniques

Fixed routing techniques assume fixed topologies and known traffic patterns. Optimal route selection is then reduced to a multi-commodity flow problem with well-defined solution techniques (18,32). Appropriate routing is generally obtained via a routing directory lookup procedure

which is fixed for a network configuration (70). A routing directory contains the link address for communicating a packet from a node to any location in the network. By searching the directory for a given destination, a cross-reference is obtained to the corresponding link for transmitting the packet. Because of their inflexibility, fixed routing techniques do not perform well in hostile environments such as experienced in wartime conditions. However, use of alternate, fixed routes provides a reasonable degree of survivability in the event of hostilities.

2.4.1.3 Split Traffic Techniques

Split traffic routing, sometimes referred to as traffic bifurcation, allows traffic to flow on more than one path between a given source and destination. Using the example provided by Fultz (32), assume that two different paths Π_1 (S,D) and Π_2 (S,D), exist. Then a packet at node S would be routed over Π_1 (S,D) with probability P and routed over Π_2 (S,D) with probability 1 - P. When compared to fixed routing, split traffic techniques maintain a better balance of traffic throughout the network, thus achieving smaller average message delays.

2.4.1.4 Ideal Observer Techniques

The ideal observer routing technique requires total knowledge of the system at each instant of time. "Each time a new packet enters a node from its HOST, its route is computed to minimize the travel time to its destination node, based upon the complete present information about the packets already in the network and their known routes" (32). Though inherent network delays make the ideal observer technique of only theoretical interest, the practical value of "total knowledge" may be useful in providing an upper bound on network performance.

2.4.2 Stochastic Algorithms

Stochastic algorithms operate as probabilistic decision rules in that routes are selected utilizing network topology, perhaps combined with estimates concerning the state of the network. These estimates are based upon statistically derived delay information communicated between interconnected nodes. The delay information is generally stored in a matrix, called a routing table, maintained at each node. A routing table is used in much the same manner as the directory is for split traffic techniques since it contains delay information to all other nodes in the network over each outgoing link. Though dynamic in nature, the frequency of table updates is a function of many complex and interrelated considerations to minimize average network traversal times. A discussion of these considerations is continued in Chapter III. The following paragraphs describe three categories of stochastic techniques which have been identified.

2.4.2.1 Random Techniques

Random routing algorithms assume that each node knows only its own identity. Each message is sent forward on a link chosen at random, hence, eventually arriving at the destination in what Davies and Barber (21) refer to as a "drunkard's walk". They proposed the introduction of a bias to guide the message roughly in the right direction, but leaving a substantial random element to cope with possible link or node failures. Although algorithms using pure random routing are inefficient, they possess a surprising amount of stability for networks having high probabilities of link or node failure (9,45,69).

2.4.2.2 Isolated Techniques

The isolated routing technique, using local delay estimates, assumes that traffic loads are approximately equivalent in both directions between any given source/destination pair. This technique is also referred to as "backwards learning" since the delay estimate to the source node is based upon combinations of transit times of messages received from the source node over the selected route. A routing table is formed at each node containing the most current estimated delay to each source. When a message is to be transmitted to a particular node, the routing table is scanned for the minimum delay path (9).

The second of the isolated techniques, called the isolated shortest queue procedure, stems from early research by Baran (10) to develop an adaptive routing method with some degree of survivability in the event of hostilities. His procedure, also referred to as the "hot potato" method, requires that intermediate nodes retransmit a packet as quickly as possible after being received. Each node in the network contains a ranked list of lines leading to neighboring nodes for every destination. Packets are directed to the highest rank free line for a given destination or, should all lines be in use, to the line containing the shortest queue (70). Though initially developed as an adaptive routing alternative for military voice communications networks, Baran's "hot potato" method is credited with stimulating the research and development of ongoing packet switching concepts (21).

2.4.2.3 Distributed Techniques

The distributed class of algorithms relies upon nodes exchanging observed delay information with each other. This approach introduces an inordinate amount of measurement information into the network and is

therefore impractical for large networks. Modified procedures have been proposed by Fultz (33) using a "minimum delay vector" and McQuillan (59) who advocates the "area approach" method.

Fultz has each node exchange a minimum delay vector to each of its nearest neighbors. They in turn update the vector with their own internal delays and pass it to each of their nearest neighbors. The permutation of transmitting updated delay vectors between nodes eventually provides each node with a matrix listing delays to all possible destinations via each outgoing circuit. Repeated updates may be prompted on either periodic or aperiodic basis.

McQuillan, on the other hand, partitions the network into disjoint areas in which a particular node exchanges information with every node within its area. Information is exchanged with adjacent areas as though each were a single node. Kleinrock et al. (59) proposes an extension of the partitioning scheme with the use of m-level hierarchical routing clusters. The objective of such a scheme is to reduce the amount of routing information that must be retained at each node for determining which link to transmit a packet on. It is anticipated that some form of the area approach will provide a logical solution to many control problems which currently exist for large scale networks.

2.4.3 Flow Control Algorithms

Congestion may occur within a network on either a global or a local basis. This congestion can be relieved by the use of a hierarchy of protocols which allow the selection of alternative actions based upon information contained in message and packet control fields. The hierarchy of protocols consists of:

1) Host-to-host (message protocols),

2) Source node-to-destination node (packet protocols),

3) Node-to-node (link protocols).

The latter of these attempts to alleviate congestion from node to node and is therefore local in nature. The other two methods are global in nature since they attempt to control traffic flow in the network. They are frequently referred to as end-to-end methods. Though others undoubtedly exist, three routing schemes are described in the following paragraphs which possess one or more of these congestion control protocols.

2.4.3.1 Isarithmic Techniques

An isarithmic network is one in which the total number of packets is held constant (20). This is accomplished by replacing data-carrying packets with dummy packets. As data is prepared for the communications process, it must capture a dummy packet in order to enter the network. Such a threshold was believed to inhibit congestion since it had been noticed that the level of traffic within a network can be expressed in terms of the total number of packets.

2.4.3.2 Buffer Storage Allocation Technique

Buffer storage allocation refers to a procedure developed by Kahn and Crowther (44) in which the source node requests allocation of message reassembly space from the destination node prior to release of the message for transmission. This procedure was proposed as a solution to prevent message reassembly lockup which occurs when nodes adjacent to the destination node become filled due to packets being rejected by the destination node. Kahn first proposed that the destination node discard packets which cannot be accepted and then notify the source node of the action. The

main problem with this approach is that it produces unnecessary duplicate packet transmissions in order to communicate a message. Advance allocation of reassembly buffers, resulting in occasional transmission delays, was concluded to be more efficient than recovering from discarded packets. Though easy to implement, neither procedure is considered adequate for the real-time or data-sharing users.

2.4.3.3 Special Route Assignment Techniques

Kahn and Crowther (44) proposed an alternative to the buffer storage allocation technique in the form of assigning special routes. The assignment of these routes by a node is based upon: 1) status information received from adjacent nodes and 2) traffic patterns encountered by the node over the past several seconds. Thresholds are established to prevent the use of alternate routes in response to rapid changes in traffic flow. This is accomplished by combining measurements on the rate of change of traffic on each path with a predefined interval of time before alternate routing can be established. As a result, changes to primary routing may occur only in response to a sustained demand for higher throughput. Table 2 provides specific properties of the flow control routing algorithm.

2.5 Discussion

In 1975, Rudin (73) proposed a classification scheme which combines routing techniques with topological categories. He states that all routing algorithms fall naturally into one of the following classifications, all but one (centralized, type 2) of which has previously been described in one form or another:

Table II. Properties of special route assignment technique (44)

1. The routing selection is performed independently by each node, based upon information received from adjacent nodes and traffic patterns encountered.
2. The algorithm attempts to guarantee that individual routing decisions possess global continuity for the network.
3. Interval (synchronous) updating of routing tables always occurs and dynamic (asynchronous) updating will occur where justified.
4. In selecting routes, the network is decomposed into a union of identical and overlapping subnetworks with separate routing computed for each subnetwork.
5. For an unloaded net, links are selected which result in transiting the fewest nodes to the destination.
6. For a loaded net, traffic is diverted from fully occupied links whenever possible.
7. Changes in routing will occur only due to the sustained flow of traffic according to a new traffic pattern.
8. Additional paths will be established for a given destination which will allow individual packets to depart on separate links.
9. Traffic flow on any link in a subnetwork may occur in only one of the two directions at a time (half-duplex operation).
10. Directions of flow may change infrequently only after passing through a neutral state for a short interval of time.
11. The maximum allowed traffic through each node in a subnetwork is regulated so as to change slowly. This provides stability and allows adjustments for increased traffic flow.
12. For each subnetwork, loops in routing are quickly detected and broken.

A. Centralized Techniques

1. Type 1 - fixed
2. Type 2 - network routing control center
3. Type 3 - ideal observer

B. Distributed Techniques

1. Type 1 - co-operative, periodic or asynchronous routing table updates
2. Type 2 - isolated, local delay estimates
3. Type 3 - isolated, shortest queue
4. Type 4 - random
5. Type 5 - flooding.

Type 2 centralized techniques use a network routing center (NRC) which periodically accepts update traffic load information from each network node. The NRC uses this information to regenerate routing tables which then remain fixed until the next update. The primary disadvantages of this technique are: 1) the single network switchpoint (NRC) where the the routing strategy can change between any two packets and 2) the constrained system behavior due to single paths for each source-destination node pair. As a result, Rudin (74) concludes that the use of the NRC approach causes instability in well-balanced networks.

Centralized routing techniques have been extensively implemented and perform well within their natural constraints. The high propensity towards total system collapse due to failure of the central control facility is an example of a natural constraint of centralized control networks. Another is the inherent inflexibility for adjustments to load variations. In general, experience has shown that they experience less delay than distributed routing algorithms under stable traffic flows

(32,35,70). The properties of a highly centralized network are well known and, thus, raise few radically new problems beyond the existing technology of computer-communications systems (43,59,74).

In evaluating the relative strengths and weaknesses of centralized and distributed routing techniques, Rudin (73) proposed a hybrid procedure referred to as "delta routing". In this scheme, topological decisions are divided into two categories. Decisions having only local network impact are implemented at the node level while decisions having global impact are implemented via the network routing control center. Though it appears to take advantage of the more favorable aspects of both classes, delta routing still suffers from the weaknesses introduced by the requirement for a central control facility.

Static (deterministic) routing strategies, exemplified by type 1 of the centralized algorithms, provide optimal routing but are based on the assumption of total reliability and fixed load patterns. In general these assumptions make the static scheme unusable except for analytical purposes. The obvious solution is an adaptive routing policy (exemplified by types 1, 2, and 3 of the distributed routing algorithms) in which changes in routing decisions are based on periodically updated information about the best routes to each destination (7). Hence, adaptive routing strategies, which take advantage of a knowledge of the current system state, have generally been used in networks with non-homogeneous hosts or large aggregates of nodes and links.

Distributed routing algorithms suffer from two major shortcomings: 1) looping and 2) a tendency to route all messages to a given destination through a single neighboring node. Looping is characterized by a message

which repeatedly traverses the same set of nodes. It may occur as a result of deficient interprocess communication (61) or unfortunate timing (7). Though loopless paths for centralized routing have been in existence for some time (15) only recently have solutions to loopless distributed routing been proposed (65).

Traffic flow measurements such as priority pricing (55) and commodity flow have become useful tools to provide optimality for a routing policy (2,30,48). These techniques, adapted from the management sciences, can be used to analyze, contrast, and improve on network routing algorithms. Simulation has also been employed to derive traffic patterns under assumed distributions. Simulation statistics have been compared with results of traffic flow measurements to determine validity of efficiency predictions for various routing algorithms (27,47).

The distributed class of routing algorithms using co-operative, periodic or asynchronous updates (type 1), possibly with some bias term, appears to provide substantial advantages over other current network routing procedures. It is the use of type 1 schemes for large scale networks which offers one of the greatest challenges to the data communications industry.

CHAPTER III

THE USE OF QUEUING THEORY TO MODEL NETWORKS

3. Introduction

Analytic methods involve schemes for load leveling and algorithms to control system saturation (19). Thus, networks must be analyzed in their entirety instead of being considered as a collection of disjoint message switches (nodes). Load leveling involves assessment of constantly changing loads and deciding upon an appropriate path for minimizing communications cost, both to the user and to the network operator. Similarly, when a message enters a network, the destination node must be determined so that goals such as economy and minimum delivery times may be met. However, a more important problem is the task of defining a general network model representative of the many intracacies of the network communications process.

The simplest network consists of a single node. The flow through the node is determined by the arrival pattern of the messages which must pass through the node and the service pattern of the node. Consider the classical example of the one-man barbershop which does not make appointments and services customers on a first-come-first-served (FIFO) basis (37). If customers arrive at a uniform rate, λ, and the barber can cut hair at a uniform rate, μ, then $\lambda \leq \mu$ means that the barber will be able to service all of his customers. The remaining case, $\lambda > \mu$, implies that customers arrive faster than the barber can cut hair and a waiting line or queue will form. Subsequently, the queue will grow without bound until the barber closes shop.

In contrast to uniformly distributed arrivals, assume that arrivals are Poisson distributed about mean λ. Similarly, assume that service is exponentially distributed around mean μ. Just as with the uniform distributions, it can be stated that for $\lambda > \mu$ the waiting line will grow without bound. Stated in network terminology, on the average, users are generating messages for delivery faster than can be delivered by the network.

The basic approach to analysis involving queuing systems has been by decomposition. In this approach, complex networks are reduced to a set of single server problems. Upon completing single server analysis, the results are synthesized to form a composite solution which reflects the properties of the original network. Justification for the decomposition-synthesis process has been provided by Jackson (41) who showed that if the Markovian assumptions of independent, exponentially distributed arrival and service times hold, and if the system is stable, each node may be treated as a separate single server queue. His research was later enhanced by Burke (13) who demonstrated that systems having exponentially distributed arrival and service times produce exponentially distributed output traffic. Burke subsequently showed that traffic flow between nodes is a Poisson process.

However, since messages maintain their length throughout the communications process, discrepancies arise between the interarrival and service times, thus violating constraints on the queuing model. Kleinrock (45), though, proposed the "independence assumption" by demonstrating that the interaction between nodes in a network having sufficient connectivity causes the service and interarrival time correlation to disappear. The

combined research of Jackson, Burke, and Kleinrock has established a viable basis for assuming Poisson arrivals and exponential service for modeling networks.

3.1 Network Characterization

One of the most accepted measures of network performance is average message delay (45,46,47,60) introduced in Chapter II. Average message delay has also proven to be the most amenable metric for mathematical analysis. Analysis based upon this metric is frequently referred to as the minimum cost routing problem (79). It provides the impetus for the theoretical characterization of the network as described in the following paragraphs.

Assume a distributed network defined by the following parameters:

a) N nodes and M links,

b) an NxN traffic matrix $\bar{R} = [r_{ij}]$ where r_{ij} represents the traffic density between nodes i and j, and

c) a cost function, $f_{ij}(y_{ij})$, relating the total cost of sending a message from node i to node j to the combined traffic loads, y_{ij}, at node i and node j.

The objective of the routing problem is to accommodate the density of the traffic matrix while minimizing the total network cost defined as $Z = \sum_{x=1}^{M} f_x(y_x)$ where x implies there exists an arc from node i to node j. The notation y_M is defined to be the sum of all point-to-point flows from node i to node j.

All inputs to a node are first placed in a service queue prior to being serviced by the node. If the link selected for transmission is busy, messages are subsequently placed in a transmission queue awaiting

transmission. For practical reasons, the lengths of service and transmission queues must be finite. Therefore, if the node is near saturation, a portion of the arriving messages will be turned away for lack of space. However, since steady-state solutions are desired, finite queue assumptions are not restrictive. Each node has an exponential service time of μ_i and operates in the full duplex mode. As a result, messages are not required to wait for a free receiving node and no blocked state is assumed.

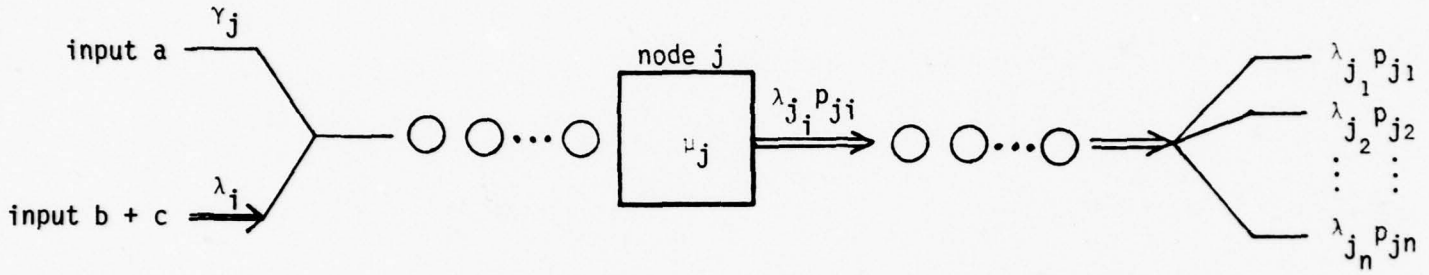
The input stream to the jth node consists of three types: Type "a" messages are those originating from users directly serviced by node j, type "b" messages pass through node j enroute to some destination node $i \neq j$, and type "c" messages are destined for a user directly serviced by j. Based upon Burke's conclusions described earlier, type "a" messages are assumed to arrive at node j from a Poisson distribution with an average arrival rate of γ_j. The arrival of type "b" and "c" messages (not necessarily Poisson (19,49)) is denoted as λ_i.

Assume for the moment that all messages have the same priority and that node j is receiving input from locally attached network users and from adjacently attached nodes. Then the true arrival rate for node j is given by (49)

$$\lambda_j = \gamma_j + \sum_{\substack{i=1 \\ i \neq j}}^{n_j} \lambda_i p_{ij}, \tag{3.1}$$

where γ_j and λ_i are as described above, n_j is the number of nodes adjacent to node j, and p_{ij} is the probability that a message is transmitted from node i to node j. Figure 6 illustrates the queue

Figure 6. Queue structure for node j



structure for the jth node with a service rate of μ_j and departure rate of $\lambda_{j_i} p_{ji}$ where double-subscript j identifies node j's type "b" and "c" message input rate to adjacent node i.

Equation 3.1 implies the existence of a network probability matrix of the form

$$\bar{P} = [p_{ij}] = \begin{pmatrix} 0 & p_{12} & \cdots & p_{1n} \\ & & & \vdots \\ p_{21} & 0 & & p_{N-1,n} \\ \vdots & & \ddots & \\ p_{N1} & \cdots & p_{N,n-1} & 0 \end{pmatrix}$$

where $1 \leq i \leq N$, $1 \leq j \leq n_i$, and $\sum_{j=1}^{n_i} p_{ij} = 1 \; \forall \; i$. The value of $p_{ij} \in \bar{P}$ is determined by the probability that a message is at node i (equal to the product of the probability that a message exists in the network and the probability that the message is being serviced by node i) combined with the cost function, $f_{ij}(y_{ij})$, of sending a message through node i to node j in route to its destination. More precisely, $p_{ij} = p_i(k_i) f_{ij}(y_{ij})$ where $p_i(k_i)$ is equivalent to

$$\frac{p(K)}{\sum_{\substack{\ell=1 \\ \ell \neq i}}^{N} p_\ell(k_\ell)}$$

and K is the N-node network state vector, $(k_1, k_2, \ldots, k_n)$, describing the number of messages in each node of the network (41). It is the comparison of various cost functions for large scale networks which receives emphasis in later chapters.

In summary, the following has been assumed:

1. Input Population: An infinite input population with a Poisson arrival rate of γ_j is assumed for messages arriving into the network at node j. Also, an arrival rate of λ_i is assumed for messages arriving from nodes connected to j. The arrival rate for the jth node is therefore described as

$$\lambda_j = \gamma_j + \sum_{\substack{i=1 \\ i \neq j}}^{n_j} \lambda_i p_{ij},$$

n_j equal to the number of adjacent nodes for node j.

2. Service Mechanism: Each node acts as a single server with service rate μ_j.

3. Queue Discipline: Entrance to each queue is FIFO. The length of the service and transmission queues combined is necessarily finite.

4. Routing Procedure: This is derived from the network probability matrix, $\bar{P}$, where $p_{ij} \in \bar{P}$ denotes the probability that a message is sent from node i to node j.

3.2 Model Description

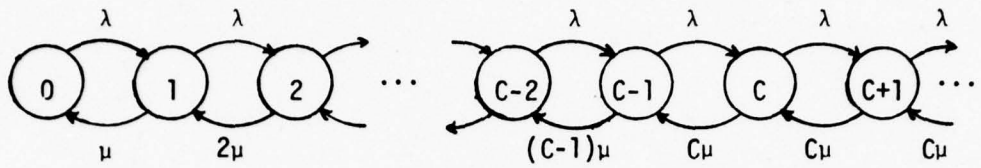
Jackson (41) defines a state variable for an N-node network as $K = (k_1, k_2, \ldots, k_N)$ where k_i is the number of customers in the ith node. Let $p(k_1, k_2, \ldots, k_N)$ be the equilibrium probability associated with state K. Jackson's theorem demonstrates that

$$p(k_1, k_2, \ldots, k_N) = p_1(k_1)p_2(k_2) \ldots p_N(k_N)$$

and that $p_i(k_i)$ is the solution to the M/M/C queuing system (using classical Kendall notation). The system descriptors for the M/M/C system are well known (37) and the solution to the M/M/C system is provided as

Appendix A. A visual interpretation of the M/M/C system is given in Figure 7.

Figure 7. State-transition-rate diagram for M/M/C (49)



An alternative and more widely accepted approach is a decomposition-synthesis as modified by Kleinrock's independence assumption. This technique allows network analysis using the basic single server queuing model (M/M/1). First, assume a Poisson distribution of interarrival times as A(t) and a distribution of service times as B(t). The expected system waiting time is derived as

$$W = \frac{\lambda \bar{t}^2}{2(1 - \rho)} + \bar{t} \qquad (3.2)$$

where λ is the average arrival rate, $\bar{t}$ and $\bar{t}^2$ are the first and second moments of B(t) respectively, and $\rho = \lambda\bar{t} < 1$ is the utilization factor. When B(t) is also exponential, the equation reduces to

$$W = \frac{\bar{t}}{1 - \rho} . \qquad (3.3)$$

Decompose the network into a series of single server problems which reflect the original network structure and traffic flow. Let Z_{ij} represent the expected delay for packets with origin node i and destination node j. Also assume an average of γ_{ij} packets per second and that $1/\mu$ represents an exponential distribution of packet lengths in bits per packet. With these assumptions, γ is defined to be the sum of the quantities γ_{ij} and

$$W = \sum_{i,j} \frac{\gamma_{ij}}{\gamma} Z_{ij}. \tag{3.4}$$

Equation 3.4 is now reformulated in terms of single channel delays. Let C_k represent the capacity of the kth channel in bits per second, λ_k represent the average packet traffic carried by channel k in packets per second, and W_k be the expected time that a packet will wait for and use the channel. In relating the λ_k to the γ_{ij} over the path selected by the routing algorithm, one can see that

$$W = \sum_{k} \frac{\lambda_k}{\gamma} W_k. \tag{3.5}$$

Using equation 3.3 with $\bar{t} = 1/(\mu C_k)$, each quantity, W_k, is determined to be

$$W_k = \frac{1}{\mu C_k - \lambda_k}. \tag{3,6}$$

Note that as $\mu C_k - \lambda_k$ approaches zero, W_k approaches infinity and, thus $C_k >> \frac{\lambda_k}{\mu}$ is necessary for W_k to be reasonable. The network has now been decomposed into a set of single-channel problems as described by equations 3.5 and 3.6. However, these equations must be adjusted to account for the general (non-exponential) packet length used by the simulation supporting the research of this dissertation.

Using the famous Pollaczek-Khinchin (P-K) formula (37) for the general service pattern (M/G/1), the average waiting time is determined as

$$W = \frac{\rho^2 + \lambda^2\sigma_s^2}{2\lambda(1 - \rho)} + \frac{1}{\mu} \tag{3.7}$$

where σ_s^2 is the variance of the service-time distribution and ρ is the utilization factor, λ/μ. If $1/\mu$ is set to be constant, the model possesses a deterministic service pattern (M/D/1) with $\sigma_s^2 = 0$. Therefore

equation 3.7 simplifies to

$$W = \frac{\rho^2}{2\lambda(1 - \rho)} + \frac{1}{\mu} . \tag{3.8}$$

The P-K results are adjusted for channel capacity, C_k, by allowing $\rho_k = \lambda_k/(\mu C_k)$ which, after the appropriate manipulation, yields average waiting time per channel as

$$W_k = \frac{2 - \rho_k}{2(\mu C_k - \lambda_k)} . \tag{3.9}$$

However, in systems with packet lengths of general distribution, average waiting time per channel is calculated with $\sigma_s^2 > 0$ resulting in

$$W_k = \frac{2 - \rho_k(1 - \mu^2\sigma_s^2)}{2(\mu C_k - \lambda_k)} . \tag{3.10}$$

This equation is necessary to account for the differences in packet lengths between user messages (528 bits), minimum delay packets (480 bits), and acknowledgement packets (118 bits). Equations 3.9 and 3.10 are proposed as approximations with the assumption that the Markovian character of the traffic flow has been preserved (31).

The effects of propagation time and overhead traffic are included by the following argument. Let $1/\mu'$ represent the average length of user traffic (excluding acknowledgements, headers, requests for next messages, parity checks, etc.) and let $1/\mu$ represent the average length of all packets. Total system delay is determined by

$$W = K + \sum_k \frac{\lambda_k}{\gamma} \left[\frac{1}{\mu' C_k} + \frac{\lambda_k/\mu C_k}{\mu C_k - \lambda_k} + P_k + K\right] \tag{3.11}$$

where K is the nodal processing time (assumed constant), P_k is the propagation on the kth channel and $(\lambda_k/\mu C_k)/(\mu C_k - \lambda_k)$ is the average time a packet waits at a node for the kth channel to become available.

In order to minimize simulation costs, certain elements of equation 3.11, in addition to packet lengths, are set constant. They are:

1. A constant nodal processing delay, K, of 10^{-3} seconds at each node traversed.

2. A channel propagation delay, P_k, consisting of eight μsec per circuit mile and a fixed modem delay of seventy μsec per channel.

3. A fixed channel capacity of 9600 bits per second.

4. A fixed node-host link capacity of 256 kilobits per second.

5. A constant user packet length, $1/\mu'$, of 528 bits.

Defining L_k to be the length of line k, total delay can now be calculated as

$$W = \sum_k \frac{\lambda_k}{\gamma} \left[0.055 + \frac{\lambda_k/9600\mu}{9600\mu - \lambda_k} + P_k + 10^{-3}\right] + 10^{-3} + 2/256000\mu \quad (3.12)$$

where $P_k = L_k(8\times10^{-6}) + 7\times10^{-5}$. (3.13)

This completes the network delay analysis using the decomposition-synthesis technique.

3.3 Developing a Cost Function

In section 3.1, reference was made to a network cost function

$$Z = \sum_{x=1}^{M} f_x(y_x),$$

where x represents connectivity from node i to node j and M is the total number of links in the network. The cost function for any particular link in the network can therefore be designated $f_{ij}(y_{ij})$ where y_{ij} represents the combined traffic loads at node i and node j. This section contains an analytical description of the cost function and its use in determining the respective values $p_{ij} \in \bar{P}$, the network probability matrix.

In determining which outgoing link to select for transmitting a packet from a node, say node i, consider the network (or graph) being converted into a tree with node i as the root. This is accomplished by eliminating all parallel paths from the network starting with a search from node i. Any node intersected by more than one path is considered to be a separate vertex within each path. Also, paths are eliminated which converge back to node i or which intersect other nodes of the same level as the current node being evaluated. Consistent with previous definitions, let n represent the maximum number of outgoing links possessed by any node in the network and n_i be the actual number of links emanating from node i. Finally, define ℓ to be the lowest level in the tree, which, for practical reasons, is constrained never to exceed four in the simulation model. Furthermore, to reduce search time, ℓ is dynamically reduced whenever a destination node is encountered prior to reaching level ℓ in a search of the tree. Label each node, starting with i = 1, using a depth-first traversal from left to right.

The cost of a path, v, whose root is node i is determined to be

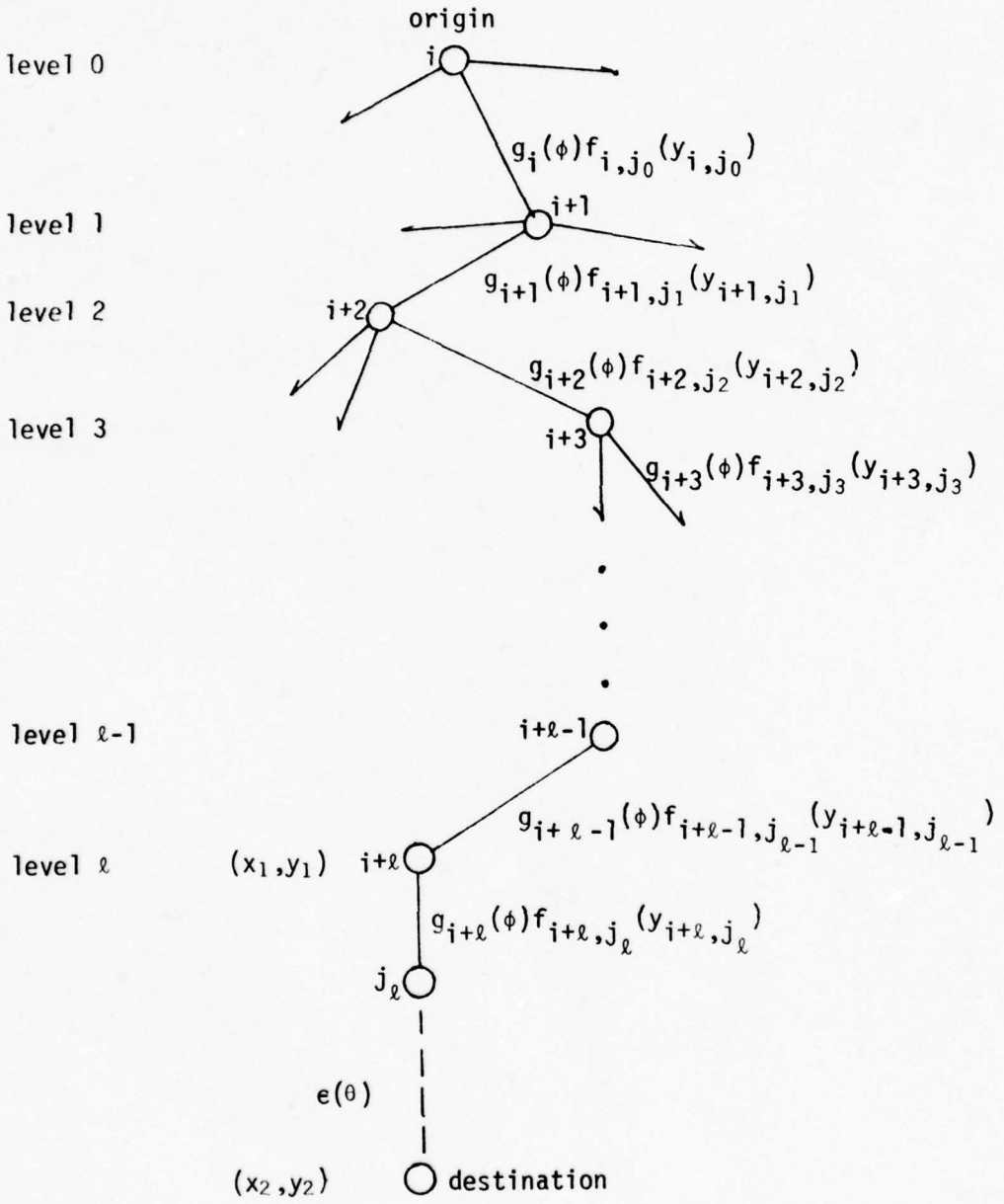
$$c_{i_v} = \varepsilon(\theta) + \sum_{k=0}^{\ell} [g_{i+k}(\phi) f_{i+k,j_k}(y_{i+k,j_k})], \qquad (3.14)$$

$$1 \leq v \leq \ell^n, \; 1 \leq j \leq n_i,$$

where ε is a function of the rectangular coordinates, θ, of the leaf node (Figure 8) and the destination node, and g is a weighting function which inverts increasing powers of ϕ depending upon the algorithm evaluated. The cost function for connectivity between i and j is defined as

$$f_x(y_x) = \alpha_x + \beta_x + \Gamma \qquad (3.15)$$

Figure 8. Tree representation of equation 3.14



where α_x = distance in miles times eight microseconds per mile propagation delay,

β_x = combined queue lengths in packets times one millisecond processing time, and

Γ = 70 microsecond modem delay.

Routing algorithms investigated by this research select for transmission the link which corresponds to the first level in a path whose cost is defined by:

$$\text{MIN } \{c_{i_v},\ 1 \leq v \leq \ell^n\}. \tag{3.16}$$

Finally, with these definitions a better interpretation of the probability matrix, $\bar{P}$, is available. Specifically, $p_{ij} \in \bar{P}$ is now proposed as:

$$p_{ij} = c_{i_j} \sum_{w \neq 0} p_w^{(i)} / \sum_{v=1}^{\ell^n} c_{i_v},\ 1 \leq i \leq N,\ 1 \leq j \leq n_i \tag{3.17}$$

where $\sum_{w \neq 0} p_w^{(i)}$ is the probability of one or more messages in the network located at node i and N, as defined in section 3.1, is the number of nodes in the network. The network cost function and corresponding network probability matrix are now explicitly defined.

3.4 The Minimum Cost Flow Problem

Attention is now turned to determining an optimal routing algorithm as a function of cost. A three phase argument is presented. First, an algorithm, B*, is defined to be a minimum cost algorithm if it selects the minimum cost path for any node j adjacent to the root node i in any tree t_i. The second phase establishes an ordered search algorithm and, subsequently an optimal search algorithm, A*, for the

optimum path of a network. Care should be taken to maintain a distinction between a path, as used in searching a tree, from path P, as used in traversing a network. The third and final phase relates the minimum cost and optimum search algorithms to derive optimum routing.

3.4.1 Determining the Path of Least Cost

Let there exist a tree, $t_i \in T = \{t_1, t_2, \ldots, t_N\}$, where T represents a set of trees constructed from an N node network as defined in section 3.3. Level ℓ_i is defined to be the maximum level in t_i and is equivalent to the search depth described in section 4.1.

Definition 3.1: For any t_i, a minimum cost algorithm, B*, is one which selects MIN $\{c_{i_v}, 1 \leq v \leq \ell^n\}$ where v is a path passing through the root node i of t_i.

Lemma 3.1: A minimum cost algorithm which always selects [MIN $\{c_{i_v}, 1 \leq v \leq \ell^n\} \forall i$] is a minimum cost algorithm $\forall t \in T$.

Proof: The proof is clear from Definition 3.1 if all t are constructed in the same manner and if equation 3.14 holds $\forall i$.

3.4.2 Determining an Optimum Traversal

The following discussion builds the basis for an ordered search algorithm which is used in determining the minimum cost path in a network. Let there exist some function $\hat{\delta}$ that could be used to dynamically order T for evaluation and let $\hat{\delta}(x)$ denote the value of the function at node x. Order all nodes adjacent to x in increasing order of their $\hat{\delta}$ values. An ordered search algorithm can be derived which next selects for evaluation that tree corresponding to the adjacent node with the smallest $\hat{\delta}$. The ordered search algorithm consists of the following steps (66):

1. Assume that a network is in state S and that a message is originated at node x_s, the root of tree t_{x_s}.

2. Compute $\hat{\delta}(x_{j_v})$, $1 \leq j \leq n_x$, where n_x represents the number of arcs emanating from x_s, j represents the nodes adjacent to x_s, and v represents the paths passing through each j.

3. Select that link for transmission which corresponds to the smallest $\hat{\delta}\forall x_j$. Map the system into state S = S + 1 by transmitting the message over the selected link and renaming the receiving adjacent node x_s, the root of a new tree, t_{x_s}.

4. If the new x_s is the destination, then the search is complete.

5. If not, go to 2.

3.4.2.1 An Optimal Search Algorithm (66)

Define $\hat{\delta}(x)$ as an estimate of the cost of a minimal cost path for a message constrained to go through node x. Also, let the function $\eta(x_i,x_j)$ give the actual cost of a minimal cost path between two arbitrary nodes x_i and x_j. If D is a set of all possible destination nodes, the cost of a minimum cost path from x_i to a destination is

$$\zeta(x_i) = \underset{x_j \in D}{\text{MIN}}\, \eta(x_i,x_j). \tag{3.18}$$

By definition, any path from node x_i to a destination node that achieves $\zeta(x_i)$ is an optimal path from x_i to a destination.

Let $\xi(x) = \eta(x_s,x)$ represent the actual cost of an optimal path from a start node x_s to some arbitrary node x. The actual cost of an optimal path from a start node x_s to a destination node x_j constrained to go through x is therefore defined as

$$\delta(x) = \xi(x) + \zeta(x). \tag{3.19}$$

Let an estimate of $\delta(x)$ be represented by

$$\hat{\delta}(x) = \hat{\xi}(x) + \hat{\zeta}(x) \tag{3.20}$$

where $\hat{\xi}$ is an estimate of ξ and $\hat{\zeta}$ is an estimate of ζ. An obvious choice for $\hat{\xi}(x)$ is given by the sum of the costs of arcs from x_s to x. This implies that $\hat{\xi}(x) \geq \xi(x)$. An estimate of $\zeta(x)$ shall be obtained by the evaluation of any heuristic information available from the problem domain. Using these definitions, let there exist an ordered search algorithm, A*, which uses equation 3.20 as an evaluation function.

3.4.2.2 The Admissibility of A* (66)

Let a routing algorithm be admissible if for any network a message always reaches its destination in an optimum path, P. The following arguments show that if $\hat{\zeta}$ is a lower bound on ζ, then A* is admissible.

Lemma 3.2: If $\hat{\zeta}(x) = \zeta(x) \forall x$, then before A* terminates and for any P from node x_s to a destination node, there exists an unevaluated (open) node x' on P with $\hat{\delta}(x') \leq \delta(x_s)$ (66).

Proof: Let P be represented by an ordered sequence, $\Pi = x_1, x_2, x_3, \ldots, x_k$, where x_k is a destination node. Now, there exists $x' \in \Pi$, since if x_k is closed, A* has terminated. By definition of $\hat{\delta}$,

$$\hat{\delta}(x') = \hat{\xi}(x') + \hat{\zeta}(x'). \tag{3.21}$$

Since x' is on P and all of its antecedents on P are closed, A* can be said to have found an optimal path to x'. Therefore $\hat{\xi}(x') = \xi(x')$ and

$$\hat{\delta}(x') = \xi(x') + \hat{\zeta}(x'). \tag{3.22}$$

Now, assuming $\hat{\zeta}(x') \leq \zeta(x')$, it can be stated that

$$\hat{\delta}(x') \leq \xi(x') + \zeta(x') = \delta(x'). \tag{3.23}$$

However, since $\delta(x) = \delta(x_s) \forall x$ on P, $\hat{\delta}(x') \leq \delta(x_s)$ as claimed by the lemma.

It is now shown that A* is admissible if $\hat{\zeta}$ is a lower bound on ζ.

Theorem 3.1. If $\forall x \hat{\zeta}(x) \leq \zeta(x)$ and if all arc costs are greater than some small number τ, then A* is admissible.

Proof: Proof is by assuming the contrary, namely that A* does not always terminate by finding an optimal network traversal. Three cases must be considered to complete the proof.

Case 1: Termination without finding the destination. Because of step 4 in the ordered search algorithm, termination cannot occur without finding the destination.

Case 2: No termination. Let there exist x_d, the destination node which is obtainable from x_s in finite steps with minimum cost $\delta(x_s)$. Arc cost $\geq \tau \Rightarrow$ there exists $\hat{\delta}(x) \geq \hat{\xi}(x) \geq \xi(x) > W\tau = \delta(x_s) \forall x$ farther than $W = \delta(x_s)/\tau$ steps from x_s. By Lemma 3.2, there exists x' on $P \ni \hat{\delta}(x') \leq \delta(x_s) \leq \hat{\delta}(x)$ and no x further than W steps from x_s is ever evaluated. Therefore, by step 3 of the ordered search algorithm, A* will select x' for transmission instead of x. The failure of A* to terminate can thus only be a result of continued reopening of nodes within W steps of x_s. Let there exist a set of u(W) nodes, called $\chi(W)$, within W steps of x_s. Assume there exists a finite number of paths from x_s to x through nodes within W steps of x_s.[1] Accordingly, $x \in \chi(W)$ can only be opened at most $\bar{\omega}(x,W) < \infty$ times so let

$$\omega(W) = \underset{x \in \chi(W)}{\text{MAX}} \bar{\omega}(x,W) \tag{3.24}$$

represent the maximum of times any one node can be reopened. Hence, all

[1] This assumption is valid because of the detect and suppress mechanism described in Chapter IV to inhibit looping.

$x \in \chi(W)$ must forever be closed after $u(W)\omega(W)$ evaluations. Since no $x \notin \chi(W)$ can be evaluated, A* must terminate.

Case 3: Termination at a destination without achieving minimal cost. Assume A* terminates at a destination x_d with $\hat{\delta}(x_d) = \hat{\xi}(x_d) > \delta(x_s)$. However Lemma 3.2 shows that just before termination there exists an open node x' on P with $\hat{\delta}(x') \leq \delta(x_s) < \hat{\delta}(x_d)$. Therefore, x' would have been selected for transmission rather than x_d, contradicting the assumption that A* terminated.

Using these results, the next section shows that a reasonable limitation on the function $\hat{\xi}(x)$ produces an optimal A*.

3.4.2.3 The Optimality of A* (66)

Consider the following restrictions:

1. When A* selects x for transmission, it has already found an optimal path to x such that $\hat{\xi}(x) = \xi(x)$.

2. When A* selects a node x, $\hat{\delta}(x) \leq \delta(x_s)$. This is true because of dynamic changes in $\delta(x_s)$ as the network goes from state to state and because of the influential power of $\hat{\xi}(x)$.

3. Assume that the difference between the estimated costs to a destination from any pair of nodes x_i and x_j is a lower bound on the true cost of an optimal path from x_i and x_j, that is, $\hat{\xi}(x_i) - \hat{\xi}(x_j) \leq n(x_i,x_j)$. According to Nilsson, this assumption will not be violated if the information used to calculate $\hat{\xi}$ is applied consistently. He appropriately refers to it as the consistency assumption.

Lemma 3.3. Given the consistency assumption and that x is closed by A* $\Rightarrow \hat{\xi}(x) = \xi(x)$.

Proof: Consider the sequence of nodes selected by A* just before closing x and assume $\hat{\xi}(x) > \xi(x)$. Now, there exists a P from x_s to x.

Because $\hat{\xi}(x) > \xi(x)$, A* did not find P. But by Lemma 3.2, there exists an open node x' on P with $\hat{\xi}(x') = \xi(x')$. The lemma is proved if x' = x but, if not,

$$\begin{aligned} \xi(x) &= \xi(x') + \eta(x',x) \\ &= \hat{\xi}(x') + \eta(x',x). \end{aligned} \tag{3.25}$$

Assuming that $\hat{\xi}(x) > \xi(x) \Rightarrow \hat{\xi}(x) > \hat{\xi}(x') + \eta(x',x)$. (3.26)

Adding $\hat{\zeta}(x)$ to both sides produces

$$\hat{\xi}(x) + \hat{\zeta}(x) > \hat{\xi}(x') + \eta(x',x) + \hat{\zeta}(x) \tag{3.27}$$

which, when applying the consistency assumption, yields

$$\hat{\xi}(x) + \hat{\zeta}(x) > \hat{\xi}(x') + \hat{\zeta}(x'). \tag{3.28}$$

This is equivalent to $\hat{\delta}(x) > \hat{\delta}(x')$ which contradicts the fact that A* selected x for evaluation when x' was available. The proof is now complete.

Lemma 3.4. If $\hat{\delta}$ is the lower bound on δ, then $\forall x$ closed by A*, $\hat{\delta}(x) \leq \delta(x_s)$.

Proof: Referring to Lemma 3.2, let x be closed by A*. Clearly, if x is the destination, then $\hat{\delta}(x) = \delta(x_s)$. Assume that x is not the destination. Since A* closed x before termination, there exists an x' on P from x_s to x_d with $\hat{\delta}(x') \leq \delta(x_s)$. The proof is finished if x = x'; otherwise x was chosen for transmission in lieu of x' so it must be that

$$\hat{\delta}(x) \leq \hat{\delta}(x') \leq \delta(x_s). \tag{3.29}$$

The optimality of A* can now be proved.

Theorem 3.2. Let there exist two admissible algorithms, A and A*, $\ni$ A* is more informed than A. Also, let the consistency assumption be satisfied by $\hat{\zeta}$ in A*. If x' was selected by A in lieu of x which was selected by A*, then $\delta(x) < \delta(x')$.

Proof: Assume the contrary ∃ $\delta(x) \geq \delta(x')$. Now

$$\delta(x') = \xi(x') + \zeta(x') \Rightarrow \zeta(x') = \delta(x') - \xi(x'). \quad (3.30)$$

A must have known that $\delta(x) \geq \delta(x_s)$ and therefore it knows that

$$\zeta(x) \geq \delta(x_s) - \xi(x). \quad (3.31)$$

This equation has a lower bound estimate of

$$\hat{\zeta}(x) = \delta(x_s) - \xi(x). \quad (3.32)$$

However, A* used the function

$$\hat{\delta}(x) = \hat{\xi}(x) + \hat{\zeta}(x) \quad (3.33)$$

in selecting x. From Lemma 3.4, $\hat{\delta}(x) \leq \delta(x_s)$∃

$$\hat{\xi}(x) + \hat{\zeta}(x) \leq \delta(x_s). \quad (3.34)$$

Therefore, whatever $\hat{\zeta}$ used by A*, it must have satisfied the inequality

$$\hat{\zeta}(x) \leq \delta(x_s) - \hat{\xi}(x). \quad (3.35)$$

By Lemma 3.3, $\hat{\xi}(x) = \xi(x)$ when A* selected x. Thus

$$\hat{\zeta}(x) \leq \delta(x_s) - \xi(x). \quad (3.36)$$

However, in selecting x', A used information about x which permitted a lower bound on $\delta(x)$ at least as large as the lower bound used by A*. Thus A* must not have been more informed than A, contradicting the assumption and proving the theorem.

3.4.3 Optimum Routing as a Function of A* and B*

B* uses c_{i_v}, $1 \leq v \leq \ell^n$, as defined in section 3.3 to determine the minimum cost path in tree t_i. The cost, c_{i_v}, however, is used only to select the node j adjacent to node i which reflects the path of least resistence to the destination. Its value also includes the cost of $\ell - 1$ levels beyond node j plus a cost factor which is a function of the coordinates of the vertex at the end of the path and those at the destination. Because of the varying load conditions in a network, the

path used in deriving the minimum c_{i_v} may in fact not be traversed beyond the adjacent node j.

Proposition 3.1. An optimum path through a network may be described as a function of A* and B*.

Discussion: First, c_{i_v} must be decomposed so that elements used in its derivation are shown to be representative of the results from applying A*. Let c'_{i_v} represent that cost portion of c_{i_v} between node i and the adjacent node j along path v. Then the actual cost, $\xi(i)$, from the start node x_s to the current node selected for transmission can be represented by $\sum_{w=1}^{m} c'_{w_v}$ where v represents an association factor between the path selected using B* and the path, P, derived using A*. Let P' be that portion of the optimum path P already traversed such that m is the number of nodes along P'. An estimate for the remaining cost of P is represented by

$$\hat{\zeta}(i) = \sum_{k=1}^{\ell} [g(\phi) f^{(v)}_{i+k,j} y^{(v)}_{i+k,j}] + \varepsilon(\Theta_v). \tag{3.37}$$

Note that g and f here are summed from 1 to ℓ instead of 0 to ℓ as in equation 3.14. This is because the cost for k = 0 is included in $\xi(i)$ as the adjacent node selected for transmission. The evaluation function used by A* is determined from equation 3.22 to be

$$\hat{\delta}(i) = \sum_{w=1}^{m} c'_{w_v} + \sum_{k=1}^{\ell} [g_{i+k}(\phi) f^{(v)}_{i+k,j} y^{(v)}_{i+k,j}] + \varepsilon(\Theta_v). \tag{3.38}$$

Briefly, it should now be apparent that the use of B* is embedded in A* whose logic and proof of optimality, extracted from Nilsson, is used to derive optimal routing from point of generation to destination.

3.5 Summary

In opening paragraphs, terminology and an example are presented as a basis for subsequent analytical presentations. Development of the analytical model is made with two arguments, one using the classical M/M/C queuing system (details contained in Appendix A) and the other using decomposition-synthesis with the independence assumption as postulated by Kleinrock. A network cost function is then used to derive the probability that a message is transmitted from node i to node j. These developments provide the foundation for final arguments which propose an algorithm, A*, for determination of an optimal path through a network. This algorithm is used as a basis for the design of the simulation model described in the following chapter.

CHAPTER IV

THE NETWORK SIMULATION MODEL

4. Introduction

One of the main reasons for using simulation is that many systems defy standard mathematical or operations research techniques for their solution. This may originate from inherent and unavoidable processes in the system to be evaluated. It may also be because of the sheer mathematical intractability of the equations which represent the system (1). Both of the problems arise in the evaluation of networks which possess schemes designed to adjust routes in response to varying load conditions. In any case, it may be desirable to use simulation to enforce results obtained by other means.

Simulation supporting the research of this dissertation was used as a vehicle to evaluate several closely related routing algorithms for large scale networks. The simulation program is based on the abstract model introduced in Chapter III and includes the assumptions made in the model's specification. It is designed so that additional routing procedures, as developed, may be simulated with minimum adjustments.

GASP-IV was chosen as the simulation language because of its simplicity, because it is FORTRAN-based, and because it is flexible and can be easily modified (68). Modifications of GASP-IV became necessary to extend the simulation effort beyond 19 nodes. The required modifications were needed because of increases in table dimensions and output format.

The network simulation program is stripped of all but the most essential data collection statements. This was necessary because of the exponential increase in computer time required to simulate increasingly large networks (Figure 9). Particular emphasis was given to program design so that calculations and subroutine calls were minimized. These design considerations combined with the use of FORTRAN H (Opt-2) reduced overall execution time by approximately 30-35%.

Though the simulation program was written for execution on the Amdahl 470v/6, only the subroutine for generating random numbers need be rewritten to be portable to other type machines. The program requires 730,000 bytes of storage for networks of 16 nodes with 150 buffers each, to 2,000,000 bytes for networks of 1024 nodes with 25 buffers each. It consists of approximately 1500 executable FORTRAN statements comprising 14 routines exclusive of GASP modules. From 125,000 to 1,400,000 bytes of memory are devoted to event and entity queues and roughly 80,000 bytes are required for network descriptor tables.

The general system flow of the simulation program is shown in Figure 10. Appendix C contains a user's guide and a listing of the program. GASP-IV and associated modifications can be obtained from the Industrial Engineering Department of Texas A&M University.

Entities within the simulator are messages which are routed from node to node by the particular algorithm being simulated. In order of priority, the three classes of messages recognized are: acknowledgement, update vectors for adaptive routing, and regular (user) messages. Each message in the simulator is described in a ten word array referred to as the messsage attributes. The second, third, and fourth entries in

Figure 9. Comparison of execution times for search depth 2

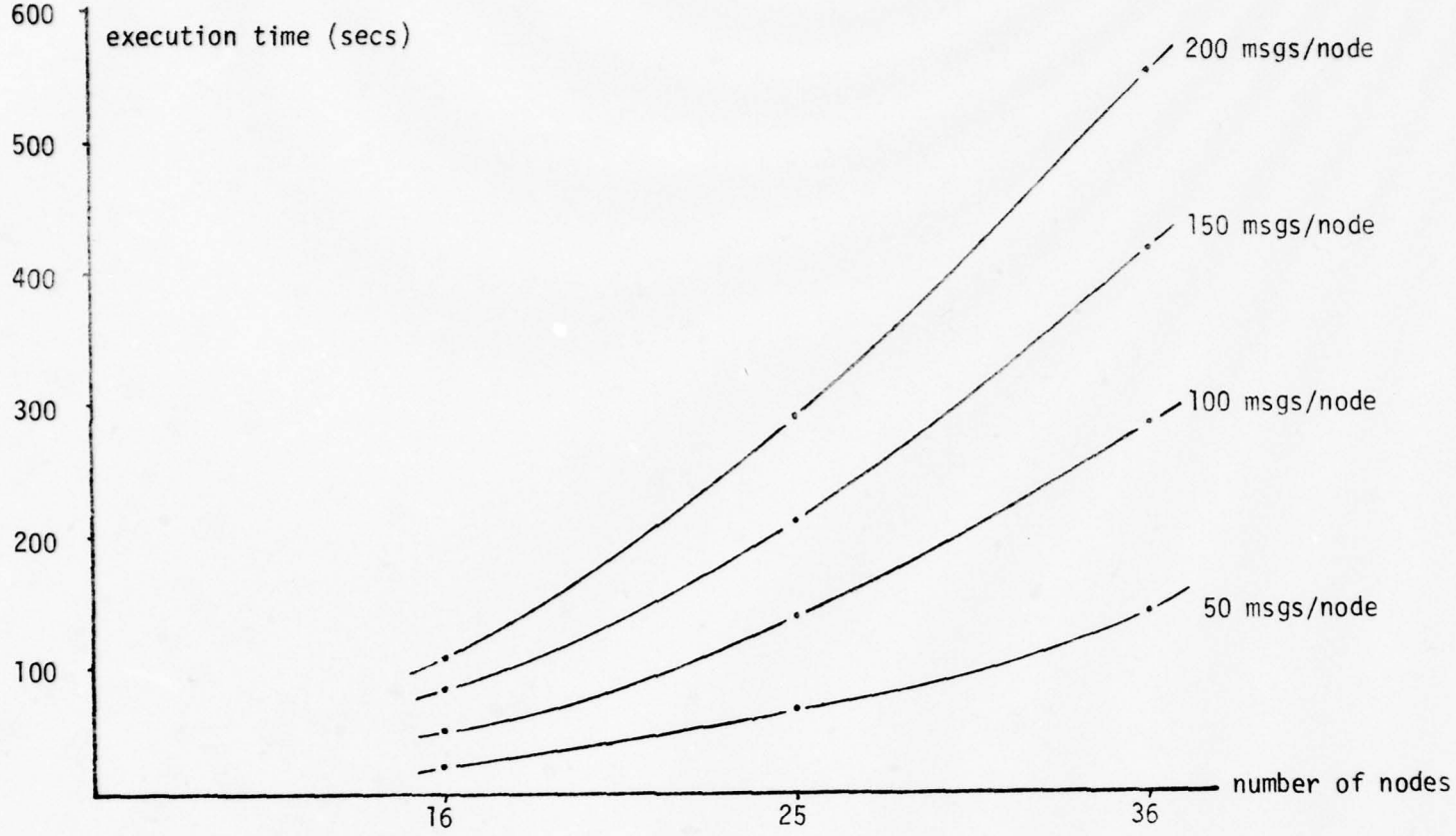
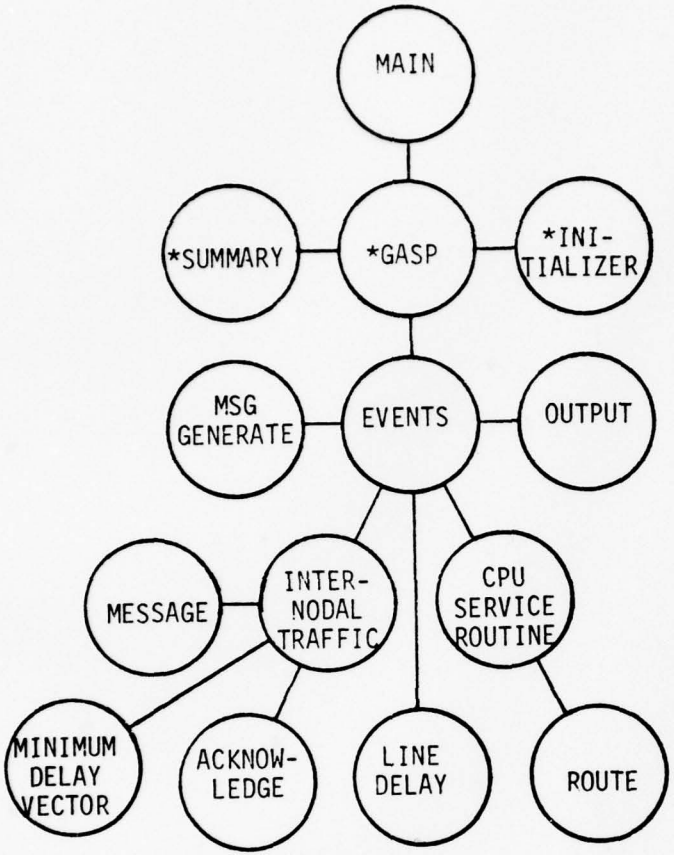


Figure 10. General system flow



* GASP IV Responsibilities

Figure 11 are graphic presentations of the array structures for the different classes of messages.

The top entry in Figure 11 is an event generation entity. It is used to identify to GASP the type of action currently being simulated, i.e., message generation, message transmission, acknowledgement, etc. Event messages are filed by priority according to designated time of occurrence as identified by the first word in the message. The simulator accumulates data by tabulating statistics on each event as it occurs.

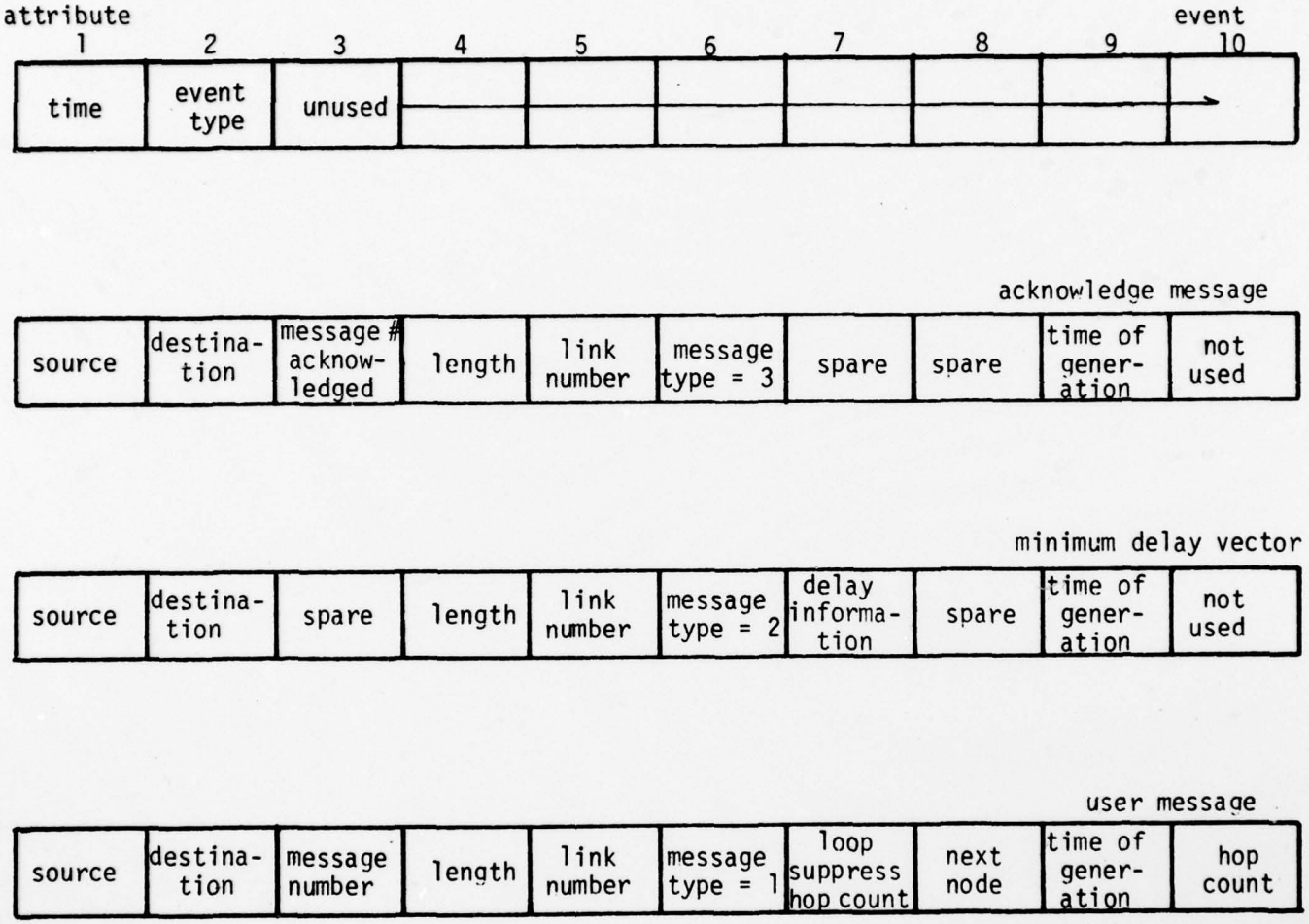
4.1 Properties of the Simulator

The network simulator is currently limited to networks of 1024 nodes or less. This constraint is imposed by the existing configuration of GASP-IV as modified in support of this research. Further increases are possible with additional modifications to GASP-IV and corresponding changes to the simulator.

Networks are generated internal to the user-provided main driver routine. The number of nodes, N, in the network to be simulated is specified through input data, although restricted to values of $N = \pi^2$, $2 \leq \pi \leq 32$. Specifying the size in this manner, these constraints provide for a programmed generation of networks possessing well distributed properties.

Each network simulated is characterized as follows. As in the abstract model, messages are assumed to arrive in a Poisson pattern. They are generated by and delivered to host computers that are attached to the network via 256 kilobit links. Each node in the network is considered to service only one host computer. Message queues at the hosts are regarded as FIFO and essentially infinite. Nodal queues are

Figure 11. GASP file entity structure (exclusive of file pointers)



ordered FIFO by message class and are finite in length. The combined length of the service and transmission queues cannot exceed a node buffer length specified in the input data. Hosts are prohibited from sending their servicing nodes a message whenever the corresponding node queues are determined to be one-half full. This ensures that a node does not become saturated due to excessive message generation by its host.

The data speed of interconnecting links between nodes is set to be constant at 9.6 kilobits per second. This speed was selected because of the ability of some modems to operate efficiently at 9600 bps with minor conditioning of voice grade circuits. However, with modest reprogramming, the interconnecting lines can be set to any other data rate or each link can be set to its own unique speed. Though links are not allowed to completely fail, they are fixed with a 10^{-5} bit error rate.

Nodes are regarded as 100 percent reliable while nodal processing delays are set constant at 10^{-3} seconds per message. Line propagation is set at eight microseconds per mile with an additional delay of 70 microseconds to adjust for modem processing.

The simulator is designed for nodes to control buffer storage, provide acknowledgements for incoming traffic, test for transmission errors, and perform message routing. No CPU intervention is required once a message is placed in the transmission queue; that is, interconnecting links are allowed to run free at their maximum data rate. This is consistent with third generation computer systems which use controllers for input/output service without CPU intervention.

Input data to the network is initialized at a load factor, RE, of 0.2 for most simulation runs. Stated in queuing terms, a load factor of RE = 1.0 implies that the arrival rate for a node is approximately equivalent to the maximum capacity of interconnecting links exiting that node, that is,

$$RE = \frac{\sum_{i=1}^{n_j} \lambda_{i_j}}{\lambda_j} \qquad (4.1)$$

where λ_{i_j}, λ_j, and n_j are as described in section 3.1. Both the initial load factor and load factor increment are provided as input parameters to the simulator.

The following list is a general description of simulator input provided by the user:

1. The number of nodes to be simulated.
2. User message length.
3. An initial network loading factor.
4. The load factor increment value.
5. The delay vector update interval in seconds.
6. The maximum search depth level.
7. The maximum queue length per node.
8. The maximum number of messages to be supported at any given time by the particular network being simulated.
9. A bias term for applying the proper weight to the function $\varepsilon(0)$ as described in section 3.3.
10. The simulator termination criteria defined as the maximum number of messages to be simulated.

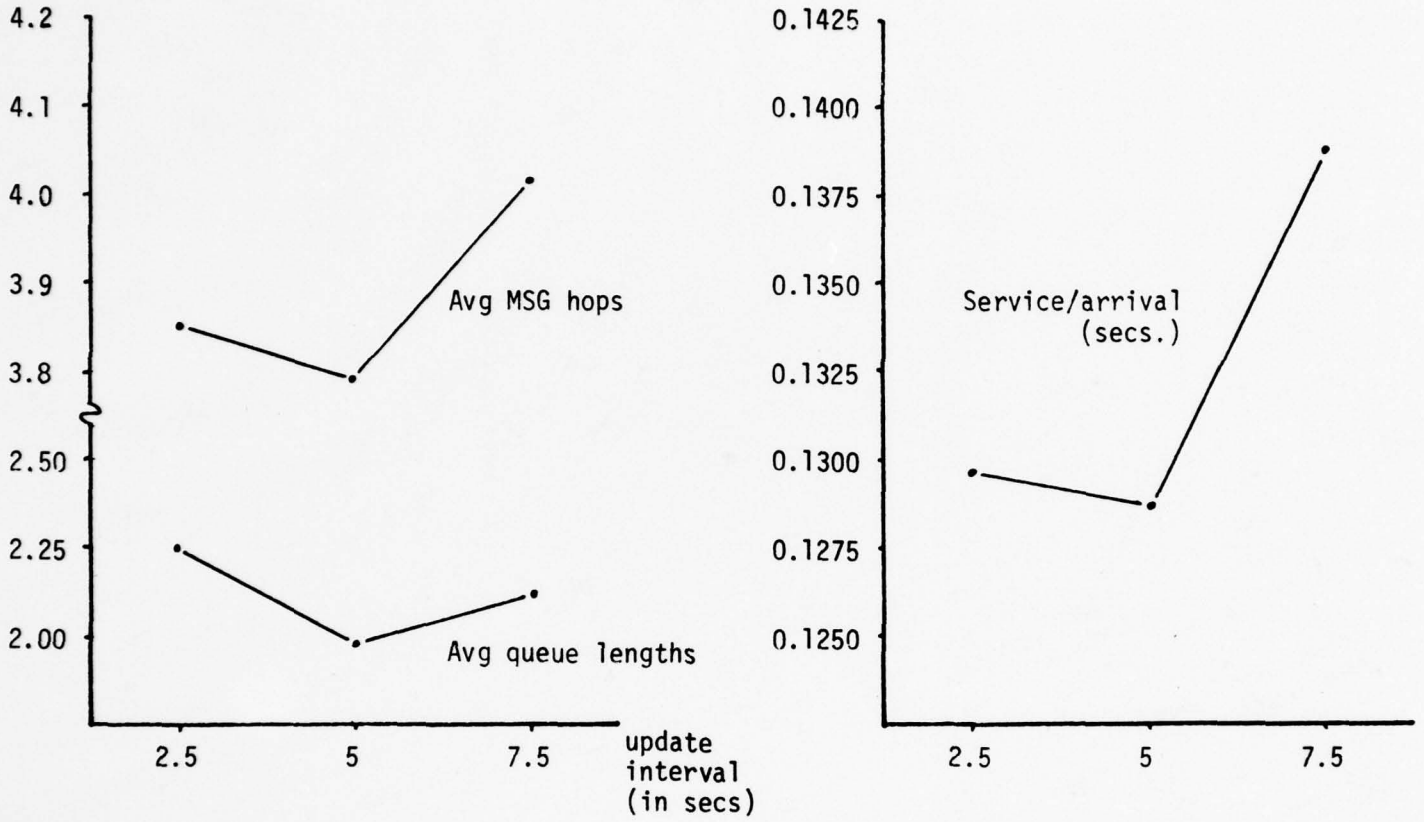
11. A print/no-print switch value for listing the network coordinates and associated descriptive matrices.
12. A value which, when combined with the output of a random number generator, expresses traffic load as a function of circuit speed. The result is used to derive the traffic mean interarrival time for each node. It is equivalent to the inverse of the arrival rate, i.e., $1/\lambda_j$ for node j where λ_j is as described in section 3.1.

The input to the simulator is used to build various topological and geographical descriptors of the network. One of these descriptors, the delay table, is particularly important to the dynamic feature of the routing algorithm. The delay table is initialized with propagation delays and later adjusted to include delays resulting from increasing queue lengths at each connecting node.

Periodically, each node in the network will generate a minimum delay vector for adjacent nodes. The delay vector contains the delay to all other nodes at the specified search depth adjusted by internal delays from the generating node. In this manner, each node is able to update its delay table with current delay information. The table only contains delay information to other nodes reachable within the specified search depth level.

The optimum update interval was determined to be five seconds during the tuning phase of the simulator (Figure 12). With exception of the 256 node simulations, all simulations were made with the five second minimum delay update interval. The update interval was relaxed to ten seconds for the 256 node simulation to allow more execution time for message routing.

Figure 12. Results of update interval analysis (load factor = 0.2)



A node interrogates the delay table upon receiving a message for which an outgoing link must be determined. The results of the interrogation are then modified according to the particular algorithm that is simulated. Finally, the outgoing link whose cost is determined to be minimum is selected for transmission. Each subsequent node that receives the message performs this evaluation until the destination node is reached. In general, the routing policy consists of identifying the adjacent node which corresponds to the minimum delay route to the destination.

4.2 General Program Flow

The user provided driver program builds the simulated network by generating rectangular coordinates, each coordinate a function of the previously generated set. The initial coordinates (node 1) are set to $(x, y) = (0, 0)$ where x and y correspond to values along the horizontal and vertical axes of the rectangular plane.

The maximum numerical value of the coordinates could have an impact on the number of overhead bits required in each message. The following procedure will ensure that the coordinate values are minimized.

1. Rotate the rectangular plane such that all points along a linear regression line representing the nodes of the network are equidistant from the x and y axes. Rotation is achieved by

$$x' = x \cos \phi - y \sin \phi$$

$$y' = x \sin \phi - y \cos \phi$$

where ϕ is the angle between the linear regression line and the hypotenuse of a right isosceles triangle with the x and y axes forming

the legs. Rotation is probably unjustified for networks whose nodes are poorly correlated.

2. Collapse the plane onto the network by subtracting the value of x' for the node closest to the y' axis (called x'_k) from all x' in the network. Correspondingly, subtract the value of y' for the node closest to the x' axis (called y'_n) from all y' in the network. The newly derived coordinates are represented by

$$(x'', y'') = (x'_i - x'_k, y'_i - y'_n), \quad 1 \leq i \leq N , \tag{4.2}$$

where N is the number of nodes in the network.

The driver program also generates topological parameters while building the network. These parameters are as follows:

1. An adjacency matrix, $\bar{A}$, whose elements, a_{ij}, are constrained by $a_{i1} = i$, $1 \leq i \leq N$ and $a_{ij} = k$, $a_{ij} \neq i$, $1 \leq i \leq N$, $2 \leq j \leq n_i$, and $1 \leq k \leq N$, where n_i, as described in section 3.1, represents the number of directed links emanating from vertex i. For a given i, a directed edge, e_i, is indicated between vertex i and vertex a_{ij}, $2 \leq j \leq n_i$. However, since all links are full-duplex, a directed edge between i and a_{ij} implies there exists a corresponding directed edge between vertex j and vertex a_{ji}, $2 \leq i \leq n_j$.

2. A distance matrix, $\bar{D}$, whose elements d_{ij}, represent the distance between node i and node a_{ij}, $2 \leq j \leq n_i$. The distance is determined by applying Pythagoras' theorem to the nodes for which connectivity is indicated.

3. A capacity matrix, $\bar{C}$, whose elements, c_{ij}, represent the capacity of the link connecting i to a_{ij}. Although all interconnecting circuits are set to 9.6 kilobits per second for this research, minor

reprogramming would provide links designated for different capacities, each unique if necessary.

4. A traffic matrix, $\bar{R}$, whose elements, r_{ij}, specify the expected traffic load from node i to node a_{ij}. These values are used to derive the mean interarrival times (see section 3.1).

Furthermore, the driver routine assigns link numbers and builds source and destination vectors that are cross-referenced with the link numbers. These vectors are valuable in reducing the amount of execution time necessary for acknowledging receipt of a packet. Control is turned over to GASP when these functions are complete.

GASP begins the simulation process by initializing internal variables and then calling a user routine INTLC to initialize user variables (e.g., the first message and the first delay update events for all nodes). INTLC is called at the start of each simulation and therefore provides for re-initializing parameters when multiple simulations are desired (68). After resetting user variables, control is passed back to GASP for the event simulation.

Event routines are required for message generation, nodal processing and routing, and line transmission. The following statistics are output at the conclusion of a simulation:

1. Rated system load.
2. Observed system load.
3. Last generated message.
4. Current simulation time.
5. Number of buffers per node.
6. Delay vector update interval.
7. Distance bias term.

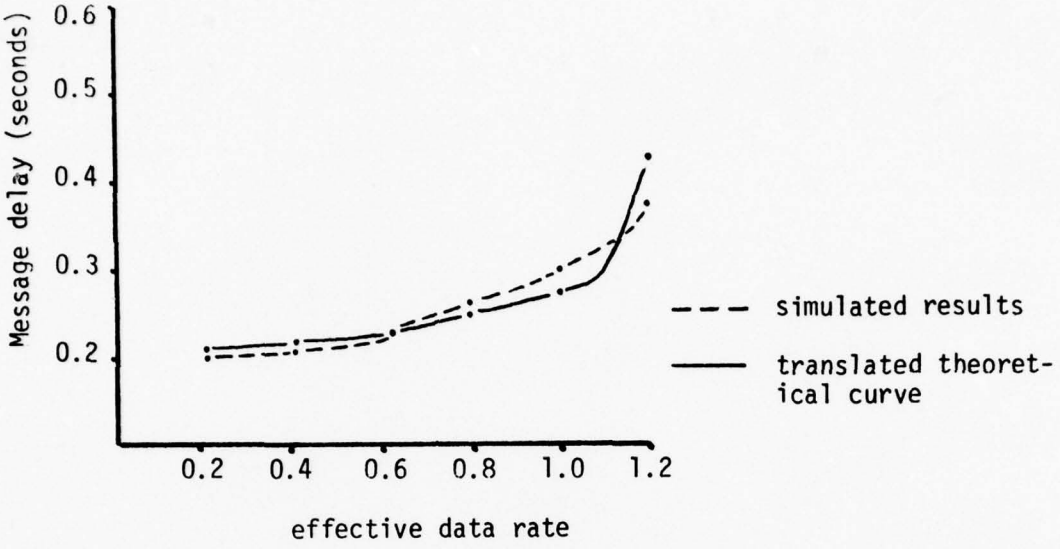
8. Rejections due to busy link.
9. Rejections due to full buffers.
10. Rejections due to busy CPU.

GASP also computes the mean, standard deviation, minimum and maximum values, and number of observations for the following GASP input initialization parameters:

1. Average message generation times.
2. Link busy time.
3. Average hops per message.
4. Average message delay.
5. Average queue length.

The observed system performance is compared to theoretical results in Figure 13. The simulation results were obtained using a 36 node network at indicated load factors. The theoretical curve, derived using equation 3.12, requires that message traces and individual nodal mean arrival rates be maintained during the simulation run. This results in exceptionally large storage and simulator execution requirements. As a consequence, an alternative method was utilized. Kleinrock's results (31) for a 19 node network are translated to a 36 node network with the equation $Y' = 5.45Y$ where Y is the mean message delay for the 19 node network and Y' is the corresponding translated result for the 36 node network. The simulation results for RE = 0.6 are used as a pivot point. A comparison of the two curves indicates that the results of the simulation have a close correlation with the translated theoretical results. This comparison is valid only if a linear transformation of average message delay is assumed between the two network sizes.

Figure 13. Simulation performance



4.3 Description of Experimental Models

Each of the curves in Figure 9 represent execution times required for delivering the number of indicated messages. An equation representing each of the curves can be obtained by using a linear combination of the well known Lagrange polynomials:

$$P_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \left(\frac{x - x_i}{x_j - x_i} \right) , \quad 1 \leq j \leq n. \tag{4.3}$$

The linear combination is defined by

$$p_n(x) = \sum_{j=0}^{n} f(x_j) P_j(x) \tag{4.4}$$

which subsequently produces

$$p_2(x) = .37x^2 - 245.03x + 144.82 \qquad (4.5)$$

using the three points given for the 50 messages delivered per node. This equation can be used to project approximate execution times for different size networks as long as all other parameters remain fixed.

The prediction for a simulation of a network of 1024 nodes at load factors 0.2 through 1.0 with an average of 50 messages delivered per node was 28.4 hours. The results in Figure 9 indicate (suggest) that execution times increase at an explosive rate for increases in the number of nodes per network. It should be apparent that such an effort is economically impractical. Therefore, a systematic six step procedure was developed using three smaller, different size networks as an inductive argument for expected results with much larger networks. The final step in the procedure consists of a sequence of simulations on a 256 node network, varying load factors between 0.2 and 1.0 at 0.2 increments.

Using an ALGOL-like syntax, the entire procedure can be summarized as follows:

```
STEP_1:  BEGIN
           ALGORITHM = 1;
           FOR N = 4 STEP 1 UNTIL 6 DO;
             *ESTABLISH THE NETWORK SIZE*
             NS = N**2;
             BEGIN
               *EVALUATE EACH SEARCH DEPTH*
               FOR SD = 1 STEP 1 UNTIL 4 DO;
               BEGIN
                 *EVALUATE NETWORK PERFORMANCE AT INTERVALS*
                 *OF MESSAGES DELIVERED*
                 FOR MPN = 50 STEP 50 UNTIL 200 DO;
                 BEGIN
                   *EVALUATE LOAD FACTORS 0.2 THROUGH 1.0*
                   FOR RE = 0.2 STEP 0.2 UNTIL 1.0 DO;
                     simulate algorithm 1 with network size =
                     NS, search depth = SD, messages delivered
                     per node = MPN, and load factor = RE;
                 END
               END
           END;

STEP_2:  Analyze data for optimum search depth.

STEP_3:  BEGIN
           *SET NETWORK SIZE*
           NS = 36;
           *ESTABLISH SIMULATION CUTOFF PARAMETER OF 50*
           *MESSAGES PER NODE*
           MPS = 50;
           *SELECT THE BEST SEARCH DEPTH*
           SD = optimum;
           *EVALUATE EACH ALGORITHM*
           FOR ALGORITHM = 2 STEP 1 UNTIL 3 DO;
             BEGIN
               *EVALUATE LOAD FACTORS 0.6 THROUGH 1.0*
               FOR RE = 0.6 STEP 0.2 UNTIL 1.0 DO;
                 simulate algorithm X with network size = 36,
                 search depth = SD, messages delivered per
                 node = 50, and load factor = RE;
           END
         END;

STEP_4:  Analyze data for optimum algorithm.
```

```
STEP_5:  BEGIN
           *SELECT THE BEST ALGORITHM*
           ALGORITHM = optimum;
           *ESTABLISH NETWORK SIZE*
           NS = 36;
           SD = optimum;
           BEGIN
             FOR MPN = 50 STEP 50 UNTIL 200 DO;
             BEGIN
               FOR RE = 0.2 STEP 0.2 UNTIL 1.0 DO;
                 establish confidence intervals at load factor
                 = RE using a 36 node network with the optimum
                 search depth and algorithm delivering 50, 100,
                 150, and 200 messages per node;
             END
           END
         END;

STEP_6:  BEGIN
           *SELECT THE BEST ALGORITHM*
           ALGORITHM = optimum;
           NS = 256;
           *SELECT THE OPTIMUM SEARCH DEPTH*
           SD = optimum;
           *EVALUATE NETWORK PERFORMANCE AT INTERVALS OF*
           *MESSAGES DELIVERED*
           BEGIN
             FOR MPN = 25 STEP 25 UNTIL 100 DO;
             *EVALUATE LOAD FACTORS 0.2 THROUGH 1.0*
             BEGIN
               FOR RE = 0.2 STEP 0.2 UNTIL 1.0 DO;
                 simulate a network of 256 nodes delivering 25,
                 50, 75, and 100 messages per node using the optimum
                 algorithm and search depth;
             END
           END
         END;
```

Three algorithms are indicated in the above procedure (steps 1 and 2). Each algorithm is identical except for the method of deriving the total delay along a path, v. For each algorithm, a different weighting function, $g(\phi)$, was employed as described by equation 3.14. The basic form of the three algorithms is best conveyed by the syntax below. Phi takes on the values of 1, 2, and e for algorithms 1, 2, and 3 respectively.

```
          BEGIN
            *ESTABLISH A WEIGHT VARIABLE FOR FINE TUNING*
            WT = 1.0/PHI;
            *SELECT THE APPROPRIATE SEARCH DEPTH*
            LEVEL = SEARCH_DEPTH;
            *EVALUATE THE PATHS IN THE TREE*
            FOR PATH = 1 STEP 1 UNTIL TOTAL_PATHS DO;
            *RETRIEVE THE NODE TO BE EVALUATED*
            TEMP_NODE = VERTEX(LEVEL,PATH);
            *EVALUATE THE COORDINATES OF THE RETRIEVED NODE*
            A_VALUE = ABSOLUTE_OF(X_COORDINATE_OF(TEMP_NODE)
                      -X_COORDINATE_OF(DESTINATION_NODE));
            B_VALUE = ABSOLUTE_OF(Y_COORDINATE_OF(TEMP_NODE)
                      -Y_COORDINATE_OF(DESTINATION_NODE));
STMT_8:     WEIGHTED_DISTANCE = FACTOR * (A_VALUE + B_VALUE)
                      + PIVOT_POINT;
            BEGIN
              *TOTAL TREE DELAY IN THE DELAY TABLE*
              TOTAL_DELAY = 0.0;
              FOR LL = 1 STEP 1 UNTIL LEVEL DO;
              TOTAL_DELAY = TOTAL_DELAY + WT * DELAY MATRIX
                      (LL,PATH);
            END
              *ACCUMULATE THE TOTAL DELAY IN A PATH *
            DELAY_VECTOR(PATH) = BIAS * WEIGHTED_DISTANCE +
                      TOTAL_DELAY;
          END;
          BEGIN
            *INITIALIZE FOR COMPARISON OF COSTS*
            MIN COST = DELAY_VECTOR(1);
            *SELECT THE FIRST PATH FOR MINIMUM COST EVALUATION*
            PATH = 1;
            *COMPARE THE COSTS OF THE REMAINING PATHS*
            FOR PTR = 2 STEP 1 UNTIL TOTAL_PATHS DO;
            IF MIN_COST GREATER THAN DELAY_VECTOR(PTR) THEN
                      MIN_COST = DELAY_VECTOR,PTR);
              PATH = PTR;
            ELSE;
              LINK = LINK_VECTOR(PATH);
              NEXT_NODE = ADJACENCY_MATRIX(CURRENT_NODE,LINK);
          END;
```

The variable WEIGHTED_DISTANCE, in statement 8, requires further explanation. Clearly, increased distances between TEMP_NODE and DESTINATION_NODE will result in a greater value for A_VALUE + B_VALUE. If WEIGHTED_DISTANCE was set equal to this sum, then for greater distances, the algorithm would lose its dynamic routing flexibility

because of the totally dominant distance factor. Not only does routing then become fixed, but some nodes in poorly developed distributed networks quickly become saturated, resulting in messages becoming trapped by loops. This looping phenomena is given extensive analysis by Neblock (65) in his research on loop-free routing algorithms.

Consider the following example to demonstrate the use of the weighted distance function. Assume a search depth of 1, and that a message generated by node 1 (Figure 14) is destined for node 7. If the dominant routing factor is always distance, then the message would quickly assume a routing pattern of 1-3-6-2-3-6-2-3-6-... where a message may not be returned to the node from which it arrived. This is because the distances between nodes 4 and 5 and node 7 are greater than those between nodes 3 and 7. Use of a function such as that contained in statement 8 places a limit on this distance value, and therefore on the impact of the distance element.

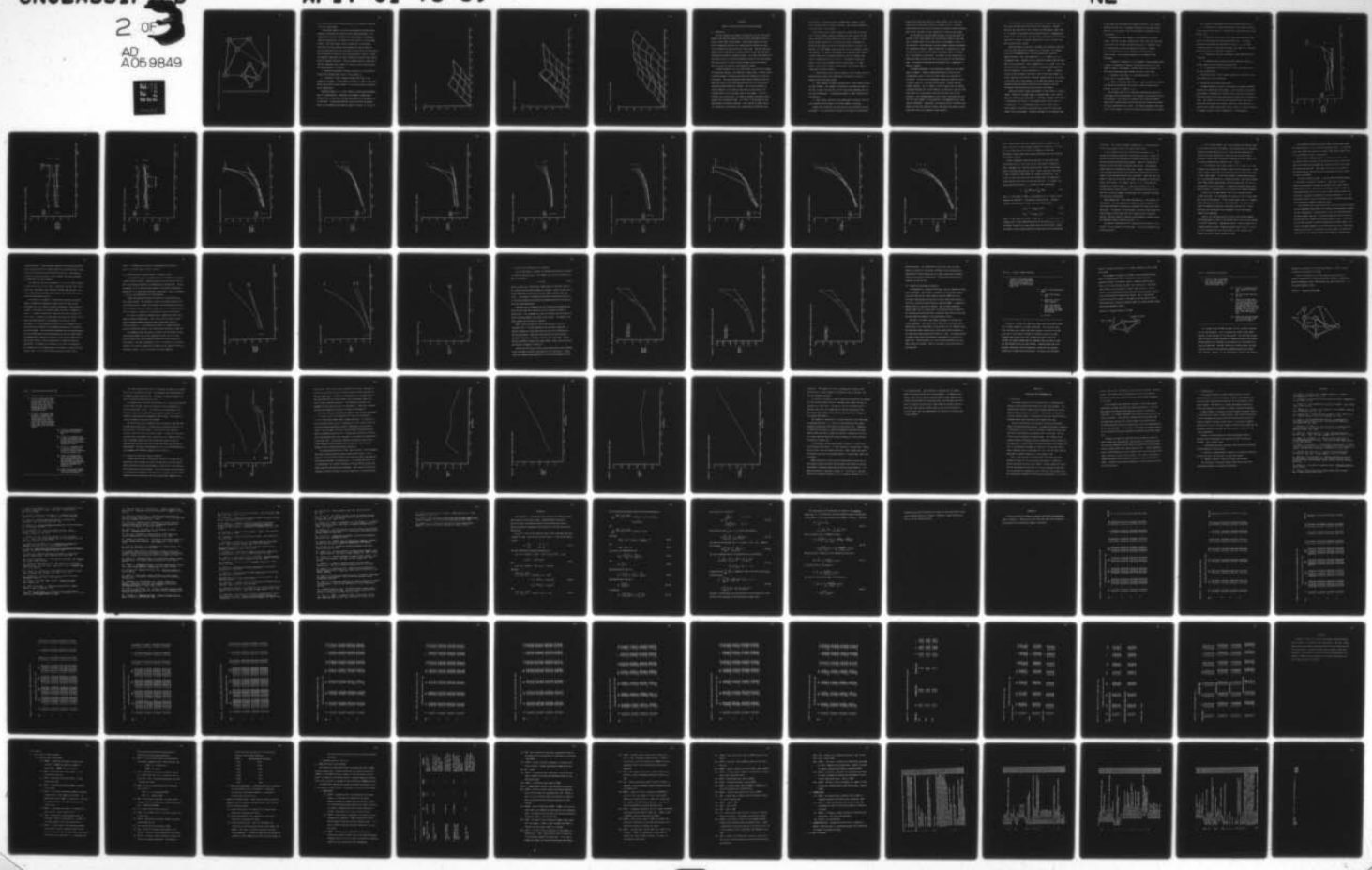
The variable PIVOT_POINT must be set to a threshold value at which point distance no longer has dominant control. FACTOR is determined by the amount of emphasis each additional distance unit should have. The specific values used for PIVOT_POINT and FACTOR were determined during the tuning phase of the simulator.

Step 3 provides for the analysis of data collected during the simulation process, and will be discussed in Chapter V. The purpose of step 4 is to add credibility to the results obtained from the network simulation. The method involves the use of estimators for the average queue length, average delay, and average system utilization. Confidence intervals are established around these estimators to reflect a percentage

AD-A059 849 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
OPTIMAL ROUTING WITHIN LARGE SCALE DISTRIBUTED COMPUTER-COMMUNI--ETC(U)
MAY 78 W H GREEN

UNCLASSIFIED AFIT-CI-78-69 NL

2 OF 3
AD A059849



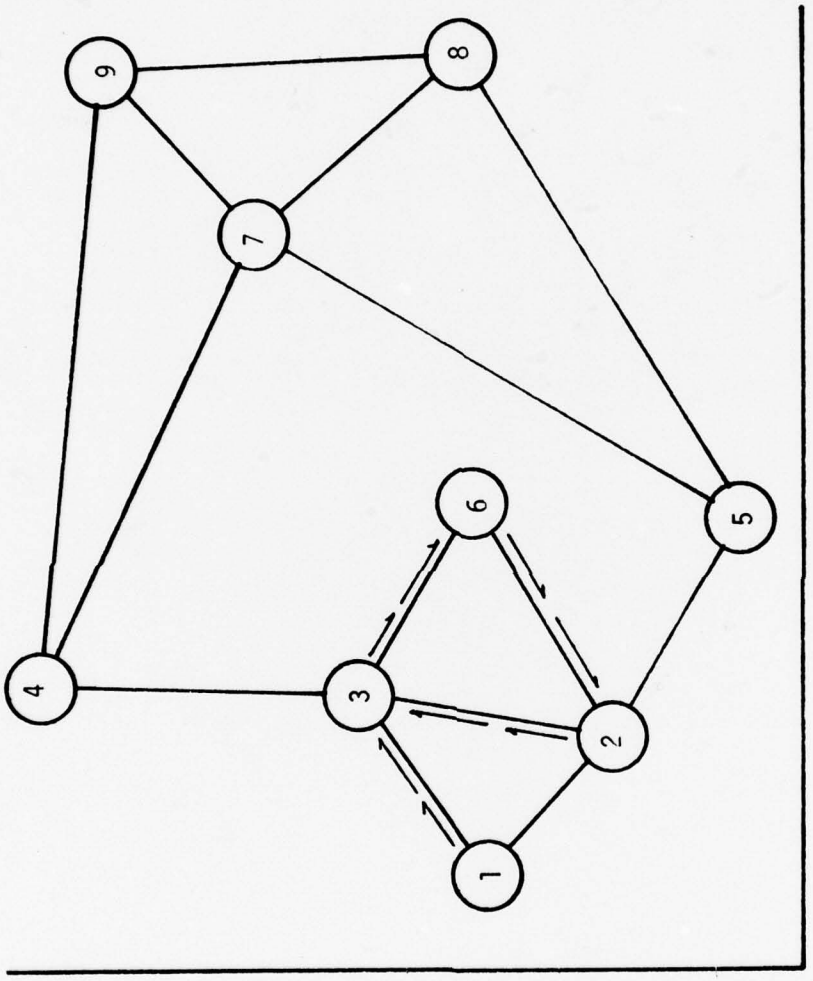


Figure 14. An ill-defined network with looping

of certainty that simulated measurements can be expected to approach the actual measurements.

Considerable emphasis was placed on developing a program which generates coordinates for nodes of a well connected network. The program had to provide coordinates with some element of randomness, establish connectivity, and conform to the criteria of a distributed network. Since the purpose of this research was to investigate algorithms for large networks which dynamically adjust routes for varying load conditions, care was taken to avoid the use of any network, such as the one in Figure 14, whose design enhances looping. "Detect and suppress" mechanisms (32) were installed to minimize looping which occurs at network saturation. Should a network possess a node that is not well connected, such as node 6 in Figure 14, one of two possible corrective measures is possible:

1. Redesign the network for better connectivity, providing for connectivity between nodes 6 and 7 in the example, or

2. Provide for unique software at nodes which have a high propensity for loops. In the network of Figure 14, nodes 2 and 3 must ensure that traffic going to nodes 7, 8, or 9 are forced into nodes 4 and 5 respectively.

Standard software is a critical factor in minimizing development costs in large networks. Therefore, the simulator assumes good connectivity as a desirable attribute when generating the networks to be simulated. The generated networks used for testing the dynamic nature of the proposed algorithms are shown in Figures 15, 16, and 17.

Figure 15. Sixteen node network

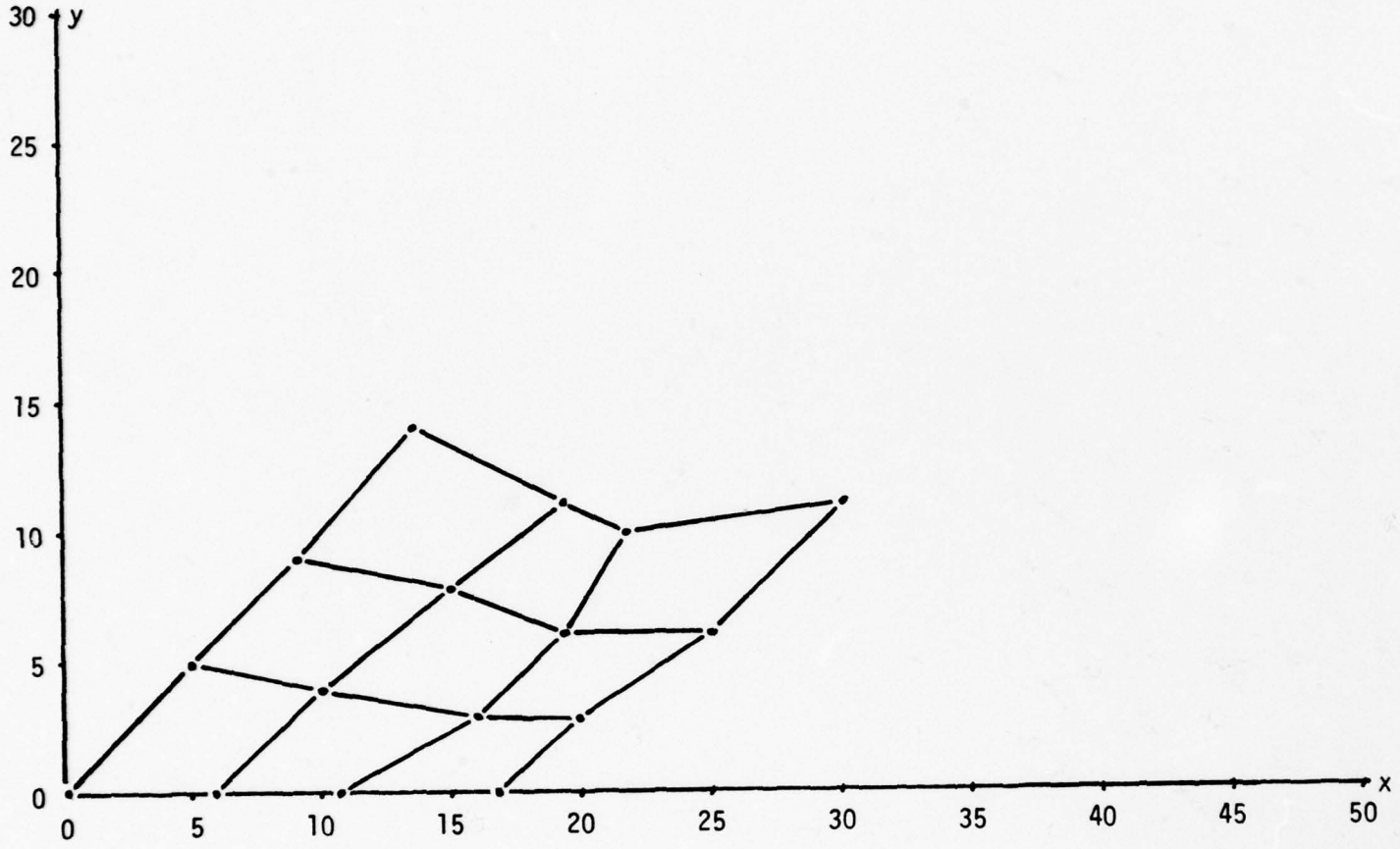


Figure 16. Twenty-five node network

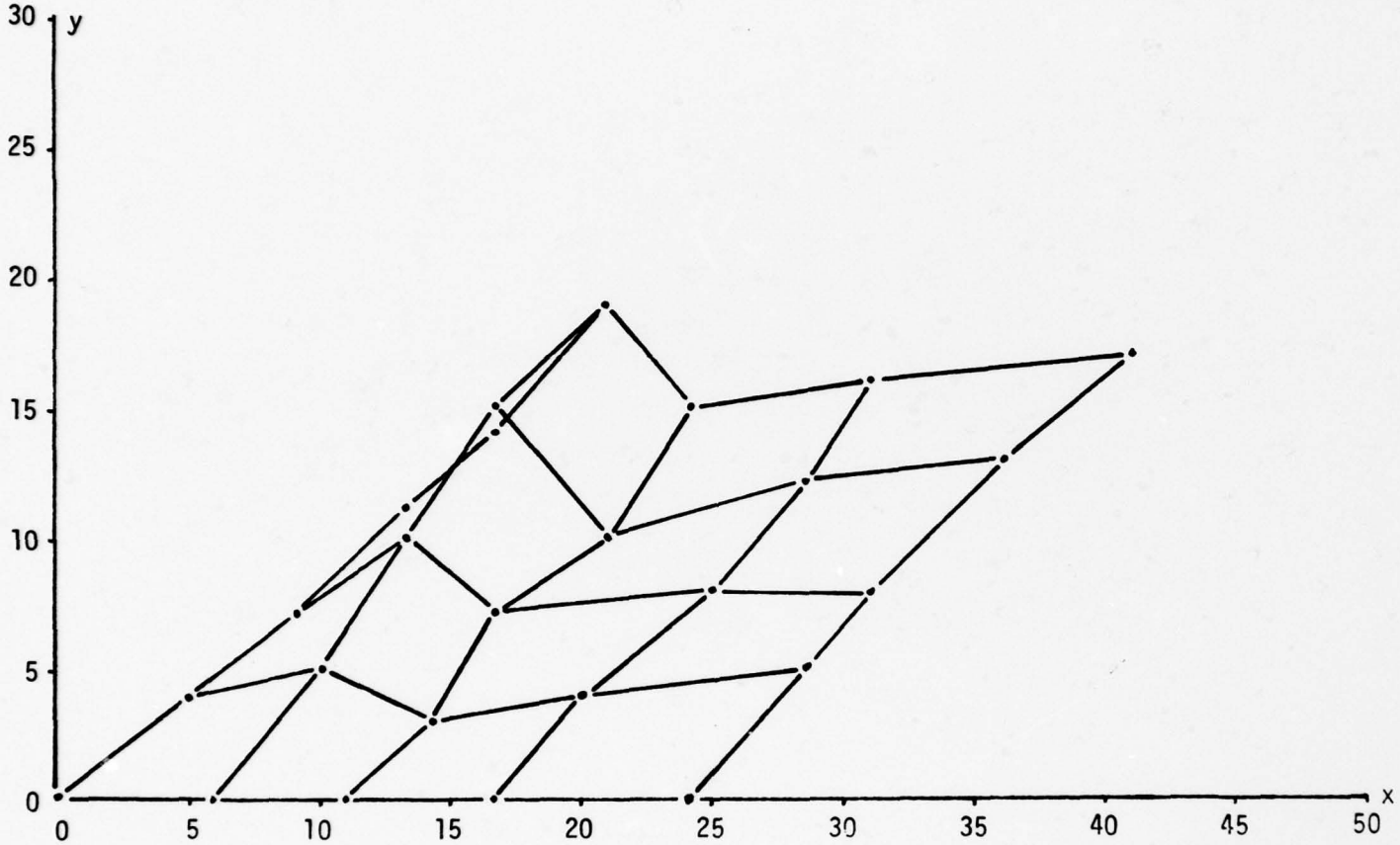
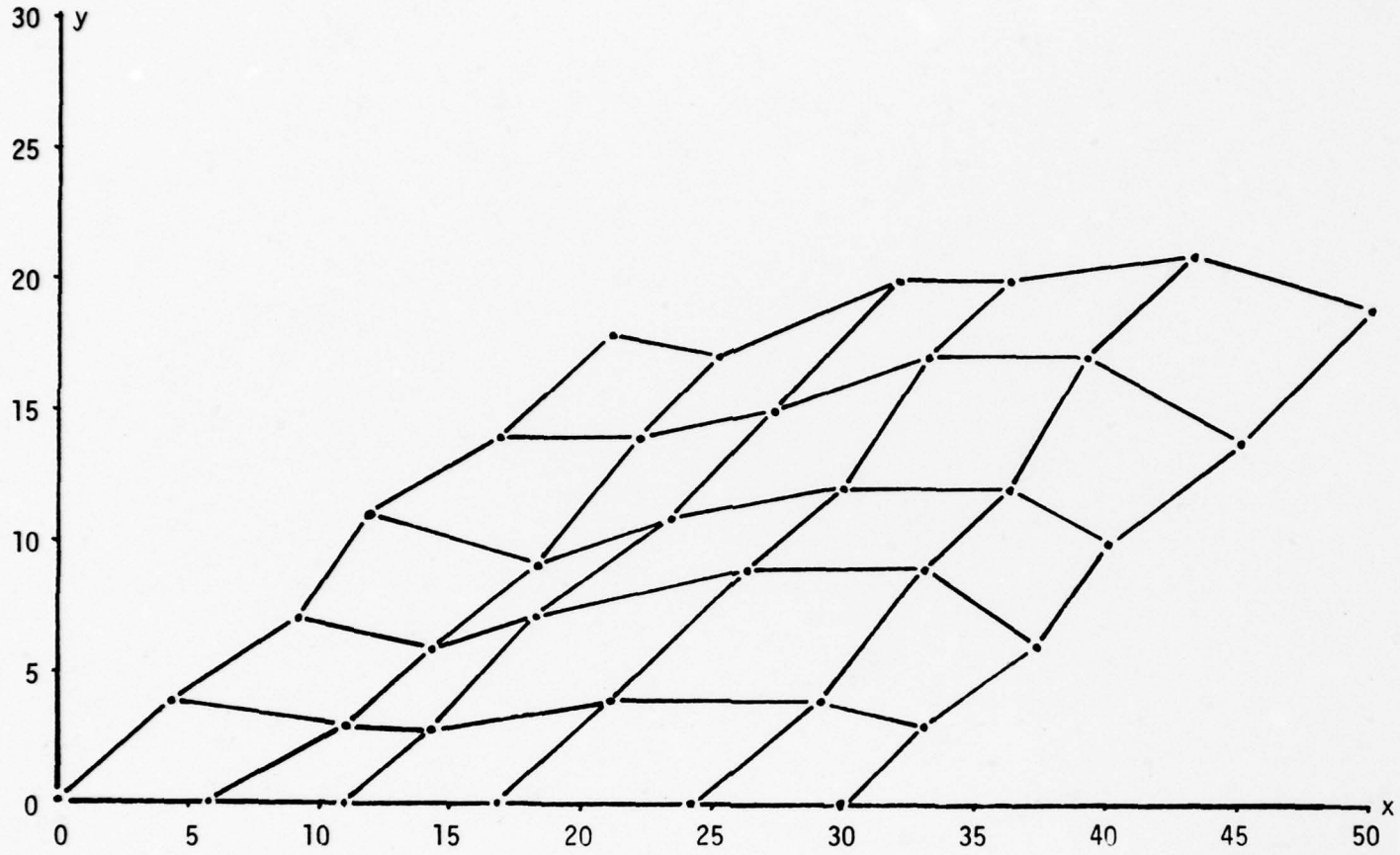


Figure 17. Thirty-six node network



CHAPTER V

ADAPTIVE ROUTING FOR LARGE DISTRIBUTED NETWORKS

5. Introduction

The most commonly used adaptive routing policies for distributed networks base decisions about routing on delay information stored in tables at each node of the network. These tables contain the most recent information available for identifying the minimum cost path to each destination accessible from a given node. They are regularly updated using a combination of delays internal to the node and delay information provided by neighboring nodes. For large scale networks of 250 nodes or more, the routing information stored at each node becomes excessively costly in terms of nodal storage required to maintain routing tables and execution time needed for updates.

A second problem, of no less significance, has to do with schemes for preventing looping. The formation of loops within a network causes a looping message to traverse more paths than necessary to reach its destination. Once loop formations have occurred, McQuillan (59) has shown that they are self-perpetuating because of the increasing and fluctuating loads placed on a network. The classical approach to inhibit looping is to include in the header of each packet a bit oriented field, one bit for every node in the network. Each bit, initially set to zero (off), is turned on whenever the packet passes through the corresponding node. This field is interrogated by each node needing to transmit a message. Ports leading to nodes with an "on" condition in the packet to be transmitted are eliminated from

consideration. A packet may become trapped when it enters a node having adjacent nodes previously traversed. The trapping phenomena is discussed later in this chapter.

Partitioned routing schemes proposed by Neblock (65) and hierarchical routing schemes proposed by Kamoun (42) both provide for loop free routing with a high propensity to avoid traps. Neblock's partitioning scheme provides for a significant savings in bandwidth by decreasing the number of bits for tracing the path of a packet. For example, a 64 node network requires 64 bits of trace overhead in global node mapping. If, however, the network is divided into eight regions of eight nodes each as proposed by Neblock, the bit-oriented trace field is reduced to $\log_2 8 + 8 = 11$ bits. The method is simple and imposes little packet overhead even for large networks. It does, however, possess the following disadvantages, some of which Neblock suggested for further research.

1. Partitioning of an arbitrary network in such a manner that the algorithm works equally well regardless of direction of flow is a complex and yet unsolved problem.

2. Minor changes in network topology have a rather dramatic effect on nodal software. For example, an increase of one node may result in the loss of one or two text bits to the trace field depending upon the existing configuration. Corresponding changes in nodal software would be necessary.

3. Delay tables residing at each node remain relatively large for large networks and fluctuate in size with changes in topology.

Kamoun uses essentially the same concept used by Neblock with two exceptions. First, partitions (clusters) are allowed to reside within

superclusters which may reside in larger clusters, etc., until the network may be completely enclosed in a global cluster. Secondly, Kamoun emphasizes the analytical approach to modeling network parameters which may be indicative of the complexity in simulating the scheme.

The hierarchical routing scheme provides a surprising amount of savings in very large networks for storing delay tables at each node. This is because there are no restrictions on the number of levels in the hierarchy. Each additional hierarchy reduces storage requirements for subordinate clusters. Adding hierarchies is effective up to some optimum level at which point storage requirements begin to increase due to the number of hierarchies. An example for the extreme case is where each node, excluding the leaf, has one and only one subordinate node. The problem of determining the optimum level is addressed in Kamoun's research.

The added advantages of the hierarchical scheme are not without impact elsewhere. Problems encountered because of the first two disadvantages noted in the partitioning scheme, are compounded in clustering. For the first disadvantage, Kamoun also mentions that a systematic means for clustering (partitioning) a network requires further research. For the second, if each cluster level has reached a maximum threshold for a given network, the addition of even one node could add a bit in the packet trace field for each level of the hierarchy. This extreme example is not a likely occurrence, but does provide a good example of the complexity in accommodating minor topological adjustments. Regardless, the research efforts of Neblock and Kamoun are bold and innovative towards developing non-looping routing algorithms that will accommodate larger networks.

A brief review of the research objectives is appropriate prior to discussing the objectives and results of the simulation. Chapter I describes the objectives of this research in broad general terms; that is, to develop a distributed routing algorithm which is independent of network size. Specifically, the algorithms investigated are practically impervious to changes in network topology with respect to packet overhead and delay table storage.

Packet overhead as required to implement the coordinate addressing technique (CAT) described in Chapter III is minimized. For example, 24 bits will accommodate over 16 million nodes with a one mile separation where x and y are bounded by $0 \leq x, y \leq 4096$ in a (x, y) rectangular plane. Eighteen bits are required to address 262,144 nodes with 10 mile separation, x and y bounded by $0 \leq x, y \leq 5120$. As a final and more practical example, 12 bits will accommodate 4096 nodes with 100 mile separation, x and y bounded by $0 \leq x, y \leq 6400$. In general, increasing the rectangular coordinate field in the packet header will allow a decrease in distance of separation between nodes or an increase in the total area of the network. As an additional benefit, the non-standard packet destination field can also be deleted since the coordinate field provides sufficient routing intelligence.

Determining nodal storage requirements for delay tables is equally simple. Establish a maximum number of adjacent nodes, say n, during design (it need not be conservative). The number of delay table entries is determined by m^n where m is the optimum search depth referred to in section 4.3. It is demonstrated later in this chapter that a search depth of 2 provides consistently good results for the three network sizes investigated. Assuming a maximum of five adjacent nodes

to each node, the delay table will require 32 entries. This feature combined with the use of rectangular coordinates to provide routing direction is the essence of the CAT which makes it adaptable to any size network.

A disadvantage of CAT, however, is the inability to prevent looping. CAT does not have a packet trace field, and thus cannot have a historical trace of nodes visited by the packet. Therefore, loops are not detectable at the exact moment of formation. However, there are methods which tend to diminish the continued occurrence of loops. The following procedure is an example of such techniques.

1. Establish a threshold, t, on the number of hops allowed before a packet is identified as looping. One obvious value for t is the number of nodes in the network. Another, used in this research, is twice the minimum path length between the two farthest nodes.

2. Maintain a hop count, c, in the packet header. If c is greater than t, determine MOD(c, t).

3. Define a secondary path limit, p. When evaluating for minimum cost traversal, discard the first choice, using the second choice path for as long as $1 \leq \text{MOD}(c, t) \leq p$.

The secondary path limit in the above procedure was tested at a value equal to the search depth in the simulator, and excellent results were obtained. Such "detect and suppress" anti-looping procedures cannot cause trapping because no restrictions are placed on the number of times a packet may visit a node. The trapping phenomena, described later in this chapter, is fatal to the efficient operation of a network.

The simulator was designed based on the following objectives:

1. To determine an optimum search depth to be used for deriving minimum cost in node traversal. Evaluate this search depth for various network sizes.

2. To determine the best of several algorithms which take advantage of information derived from meeting objective 1. The three algorithms investigated are referred to as CAT_n, $1 \leq n \leq 3$, where increasing values of n represent bias terms, $g(\phi)$ equal to 1, 1/search depth level, and 1/e as defined in Chapter III.

3. To establish a level of confidence in data obtained from the simulation.

4. To determine how the best of the CAT algorithms compare to several common earlier defined routing policies.

5. To demonstrate the operation of a large scale network using the best evaluated CATs.

Remaining sections of this chapter elaborate on results of using these objectives in the simulation.

5.1 Determining the Optimum Search Depth

An upper bound was necessary on the amount of storage considered practical for maintaining delay tables. For that reason, the maximum search depth investigated was limited to four, requiring a maximum of 1024 delay table entries where each node is allowed up to five adjacent nodes. The simplest of the three CATs evaluated, CAT_1, was used in determining the optimum search depth. Results of the simulations to determine the optimum search depth (SD) are given in Figures 18 through 29 (supporting data is given in Appendix B). Interconnecting

Figure 18. Average hops per packet, 16 node network

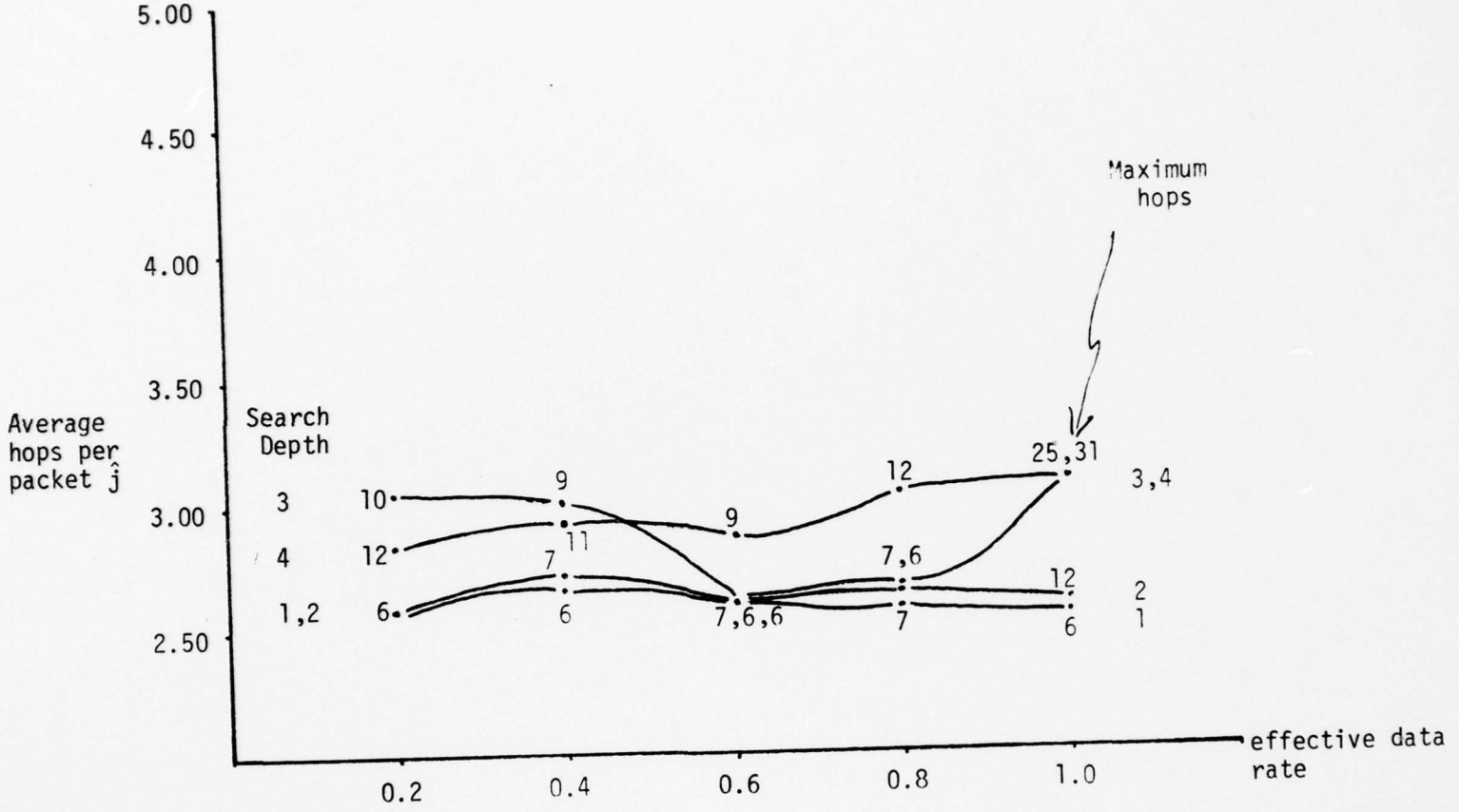


Figure 19. Average hops per packet, 25 node network

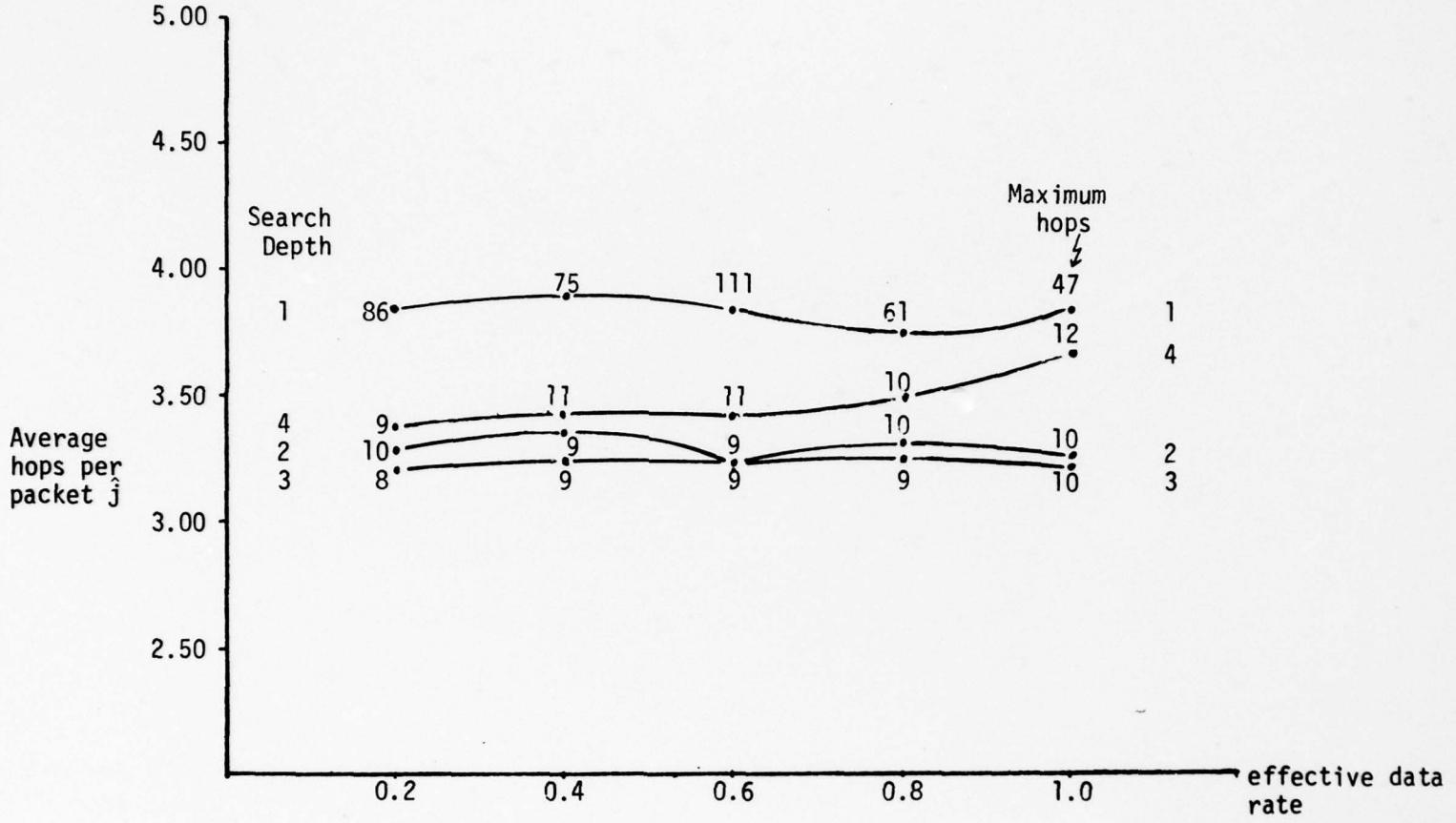


Figure 20. Average hops per packet, 36 node network

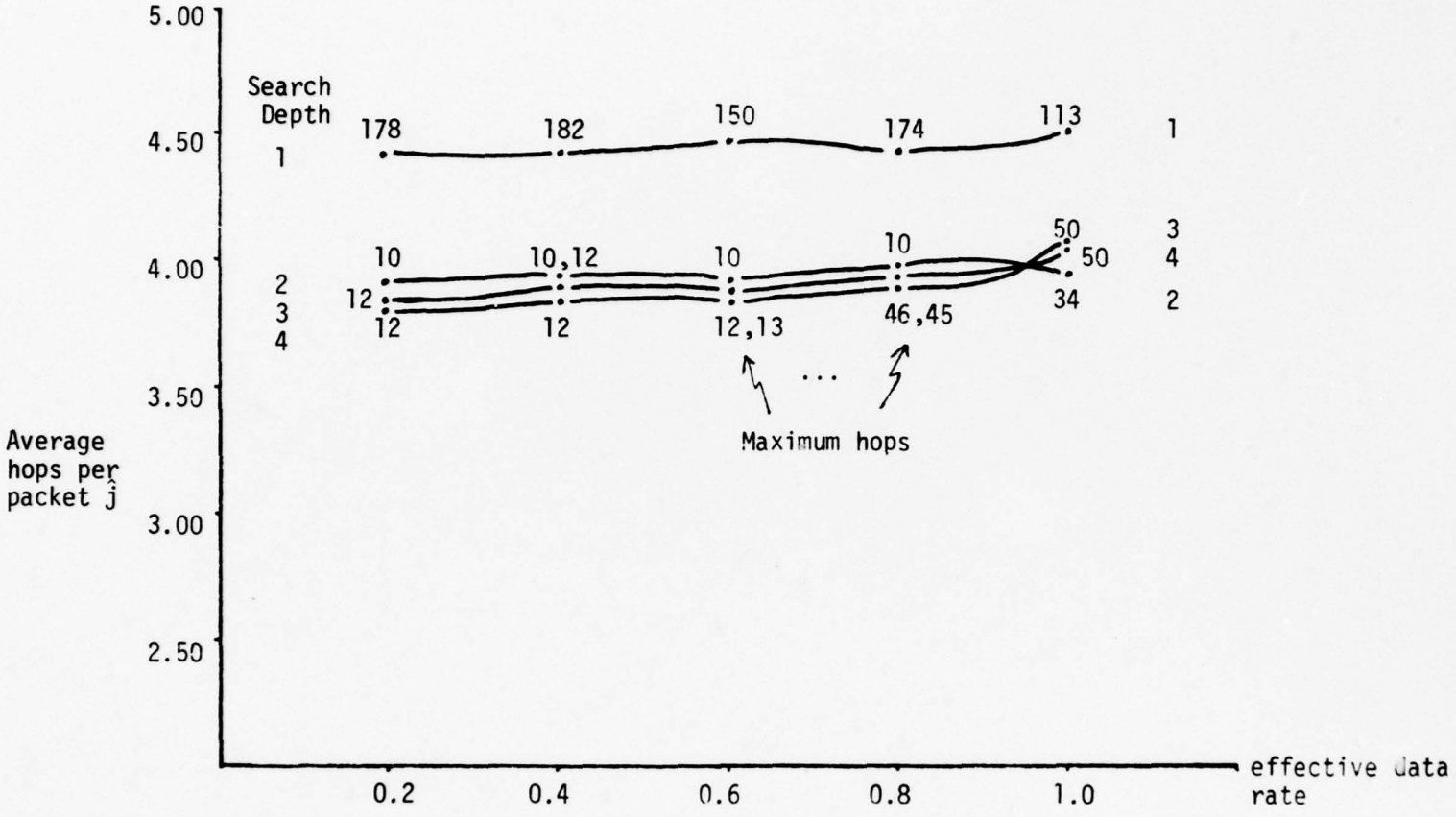


Figure 21. Average queue length for 16 node network

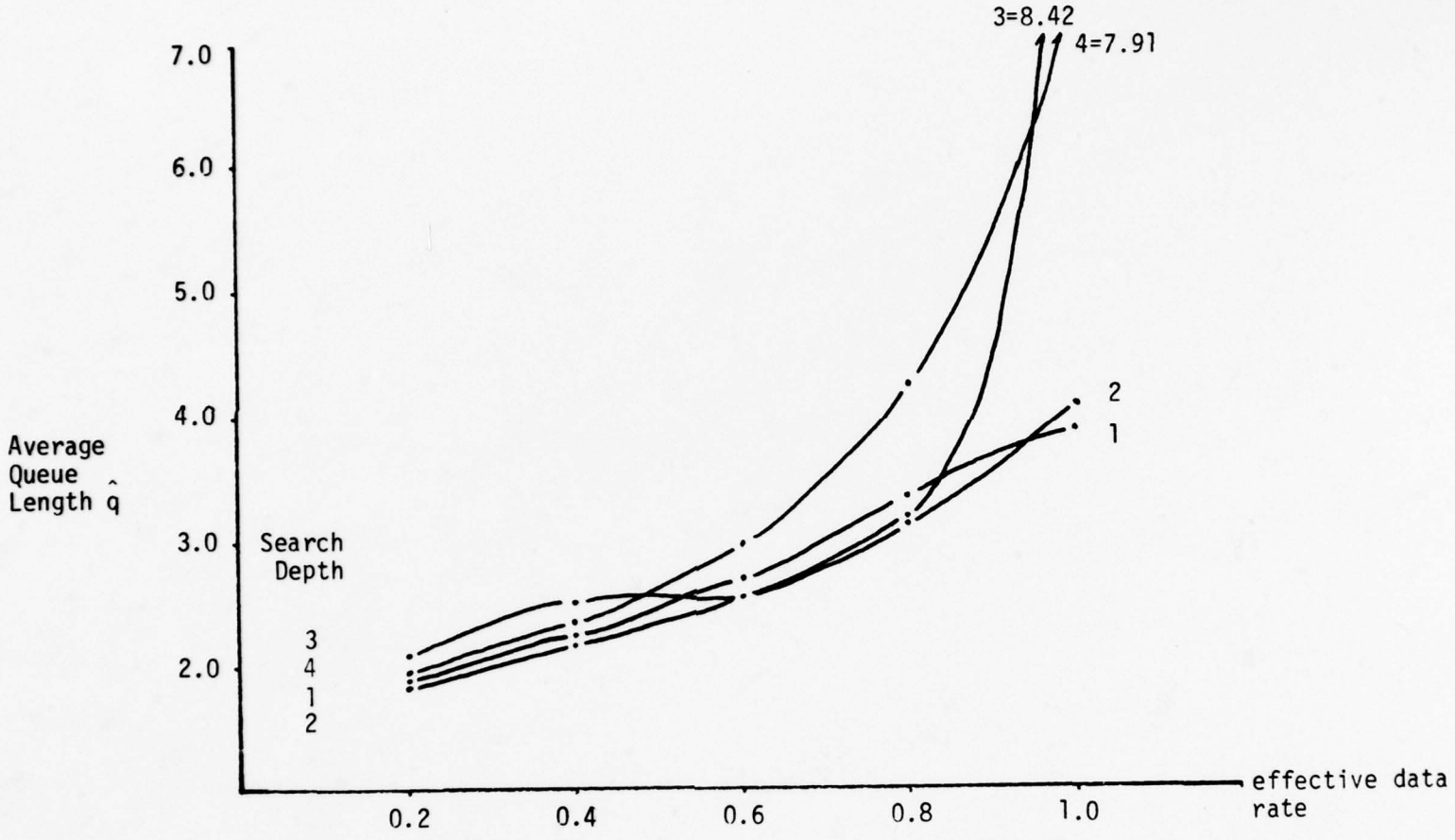


Figure 22. Average queue length for 25 node network

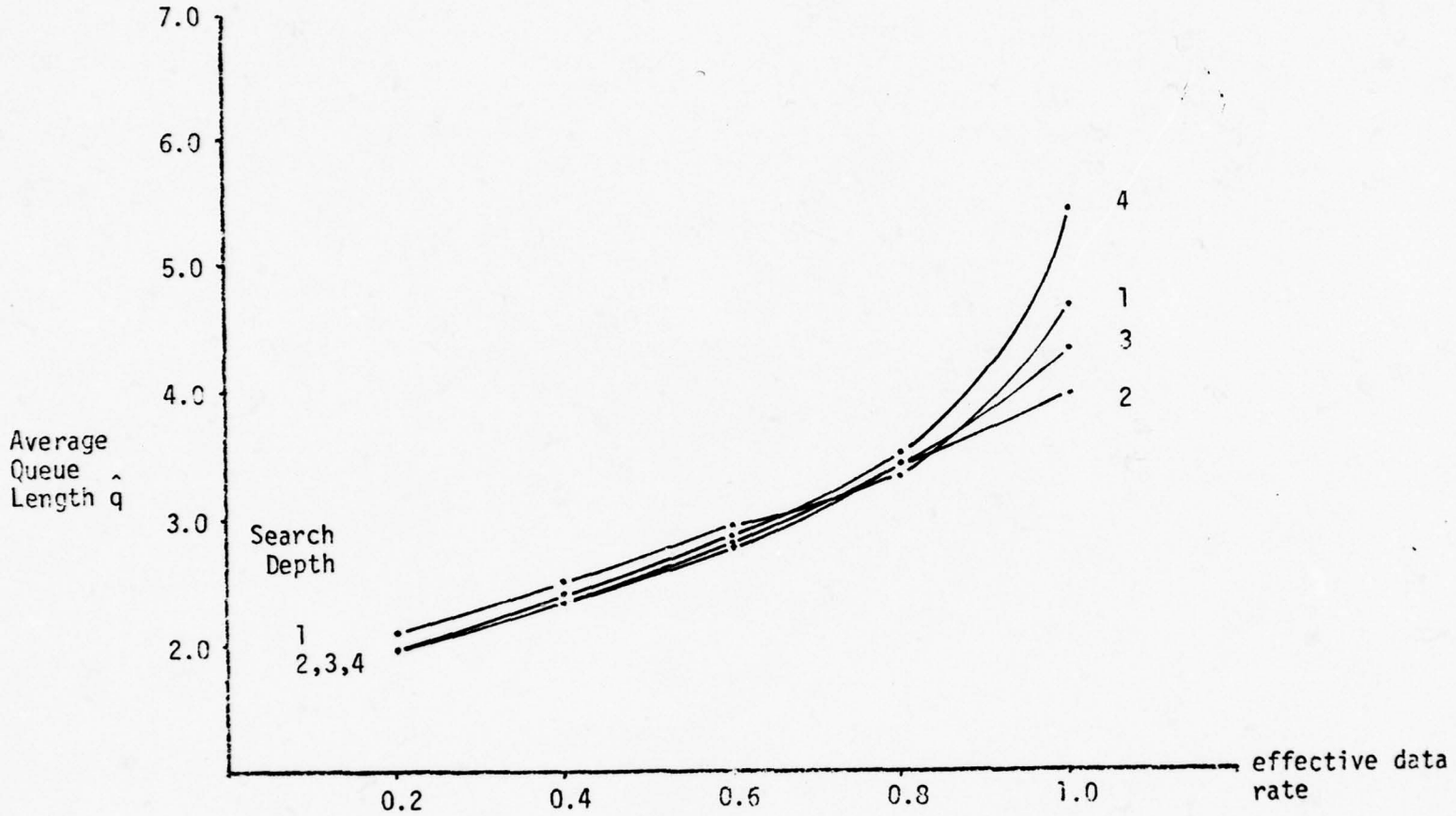


Figure 24. Average delay for 16 node network

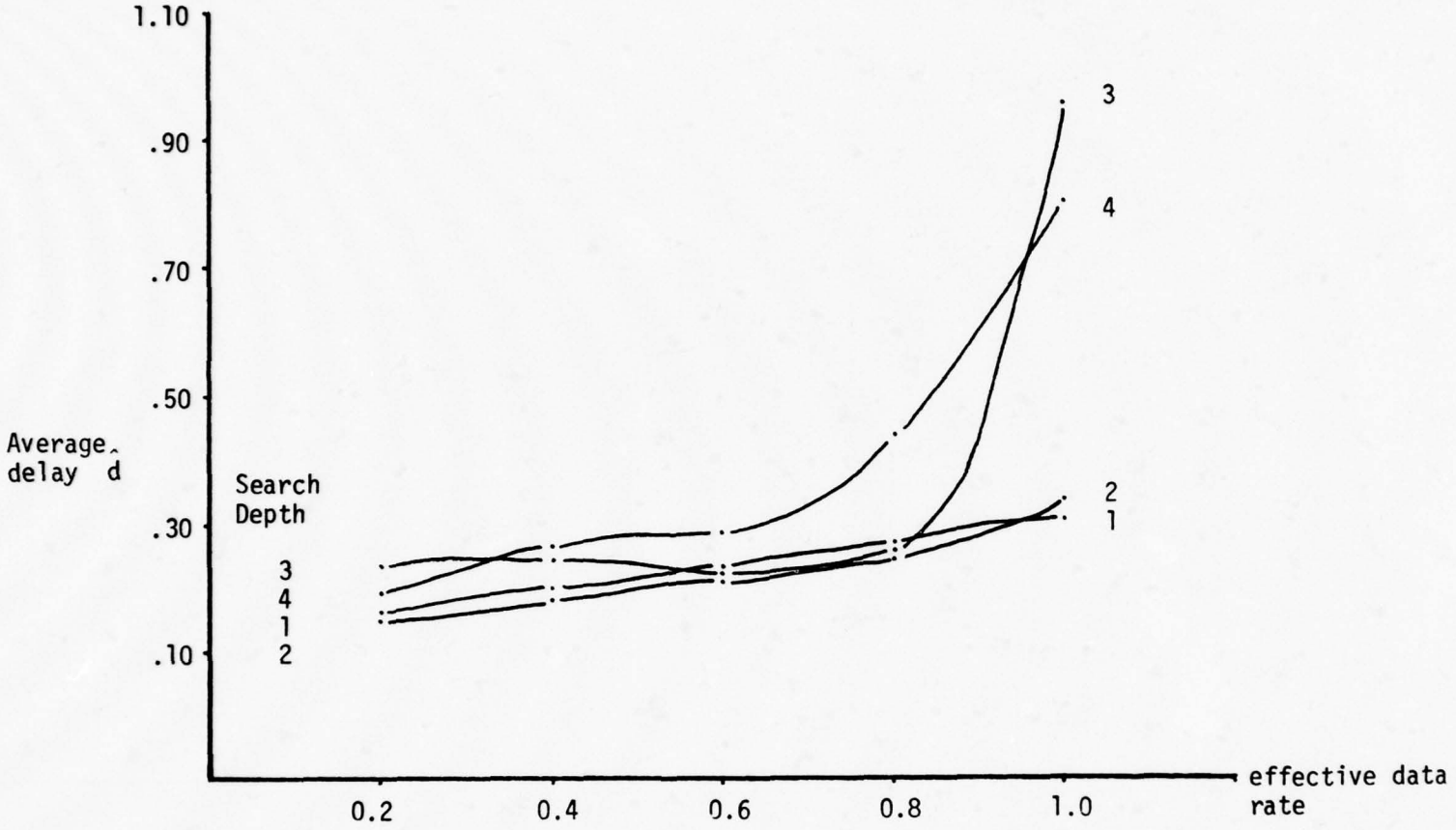


Figure 25. Average delay for 25 node network

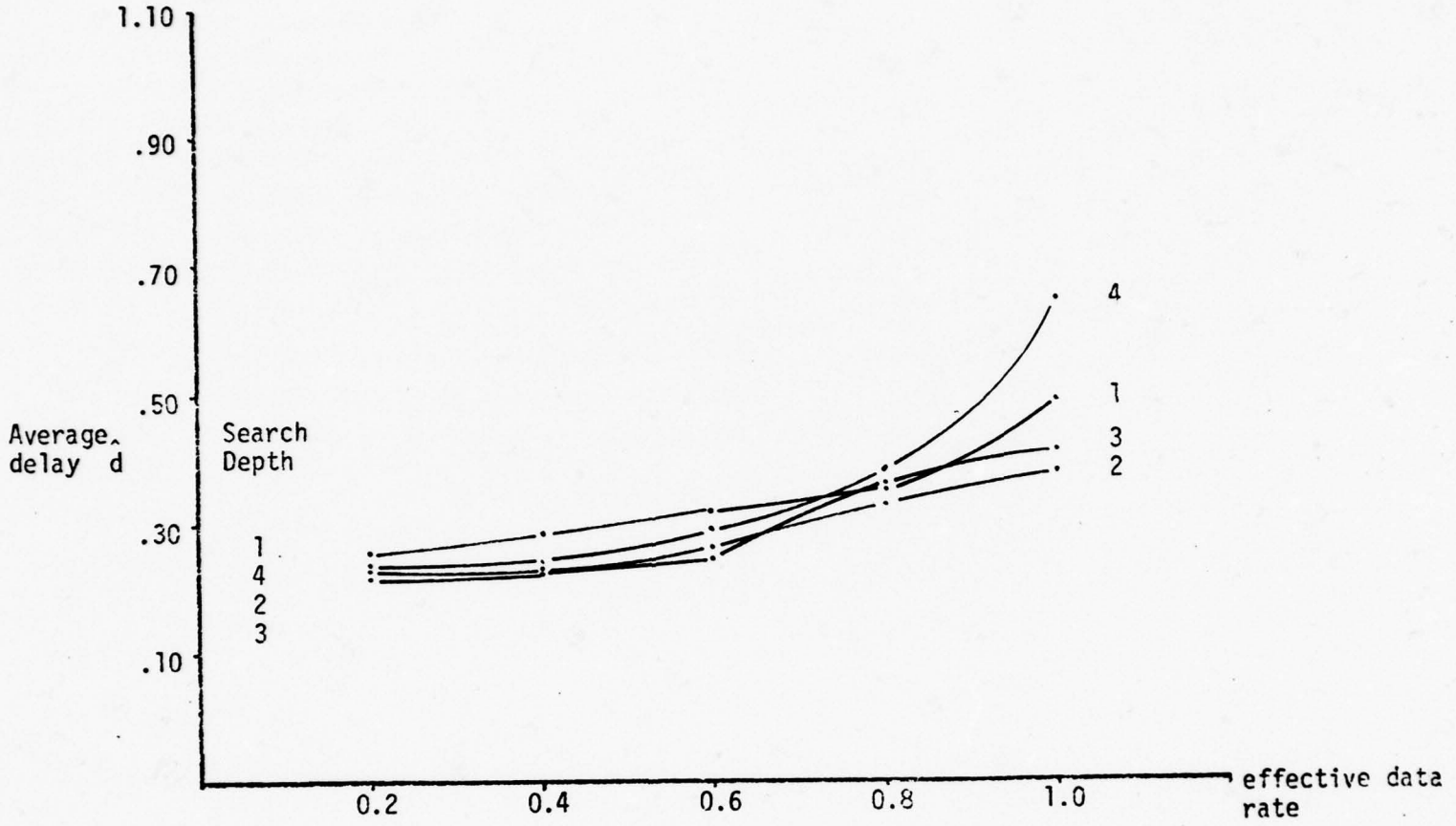


Figure 26. Average delay for 36 node network

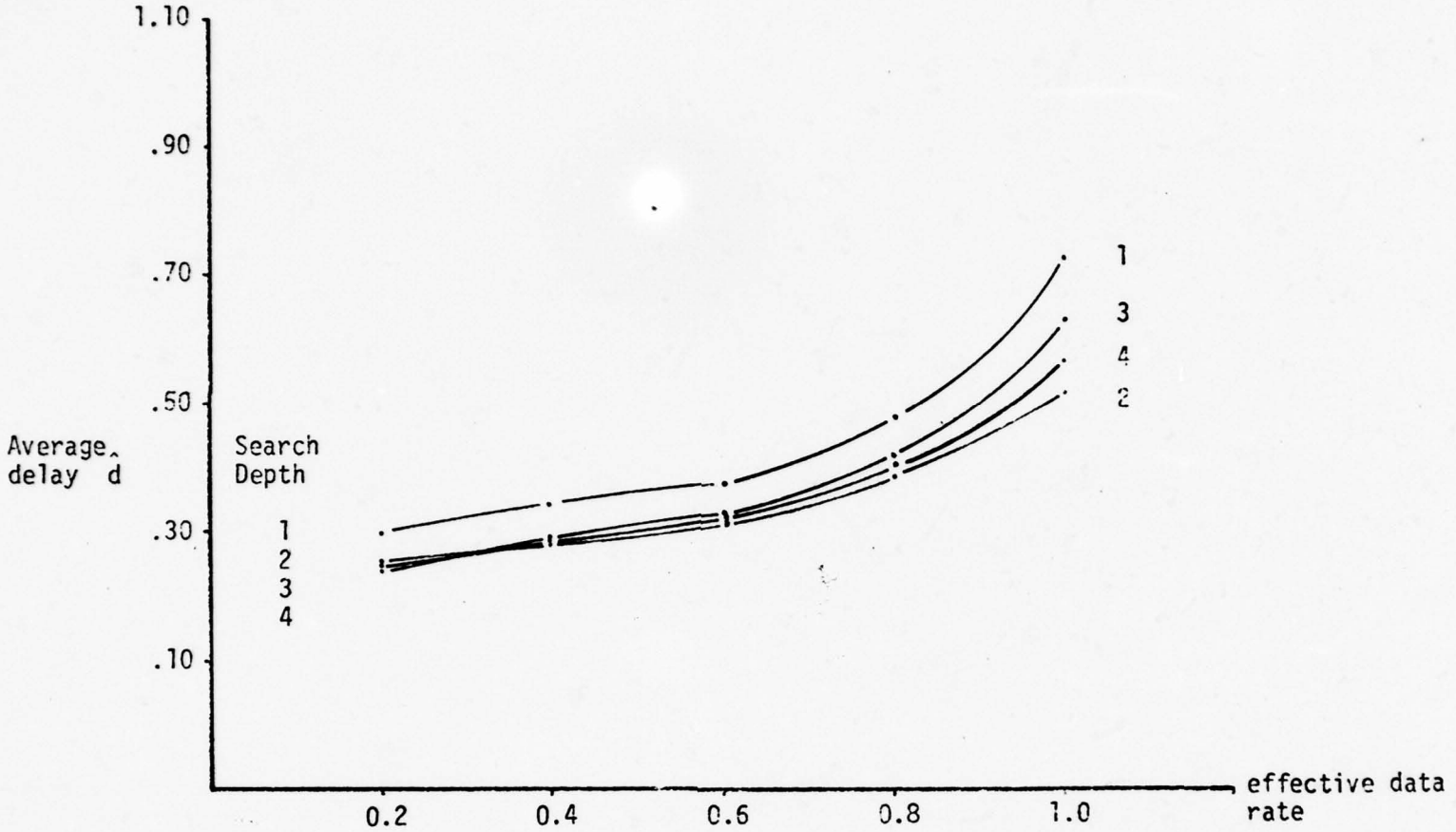


Figure 27. Utilization for 16 node network

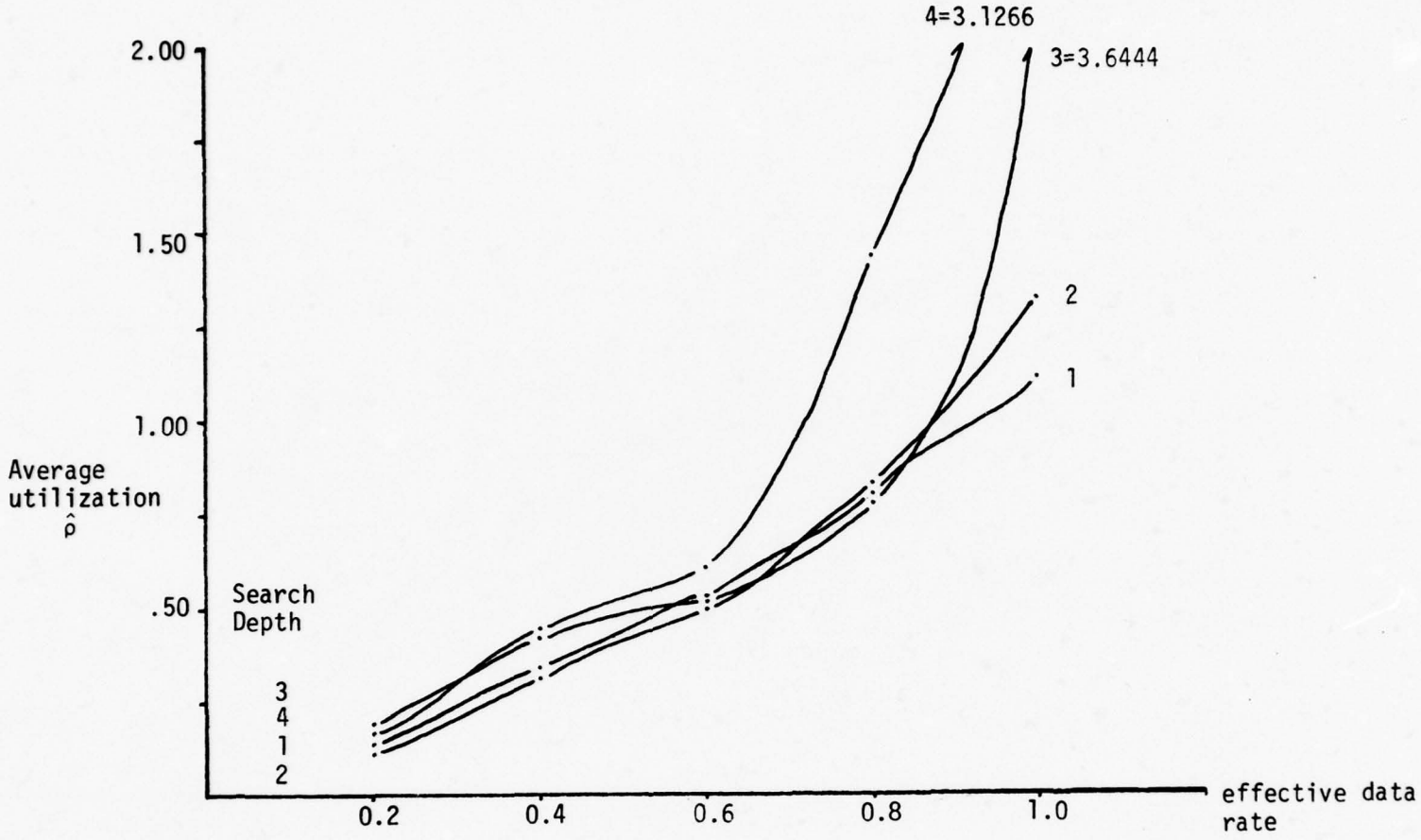


Figure 28. Utilization for 25 node network

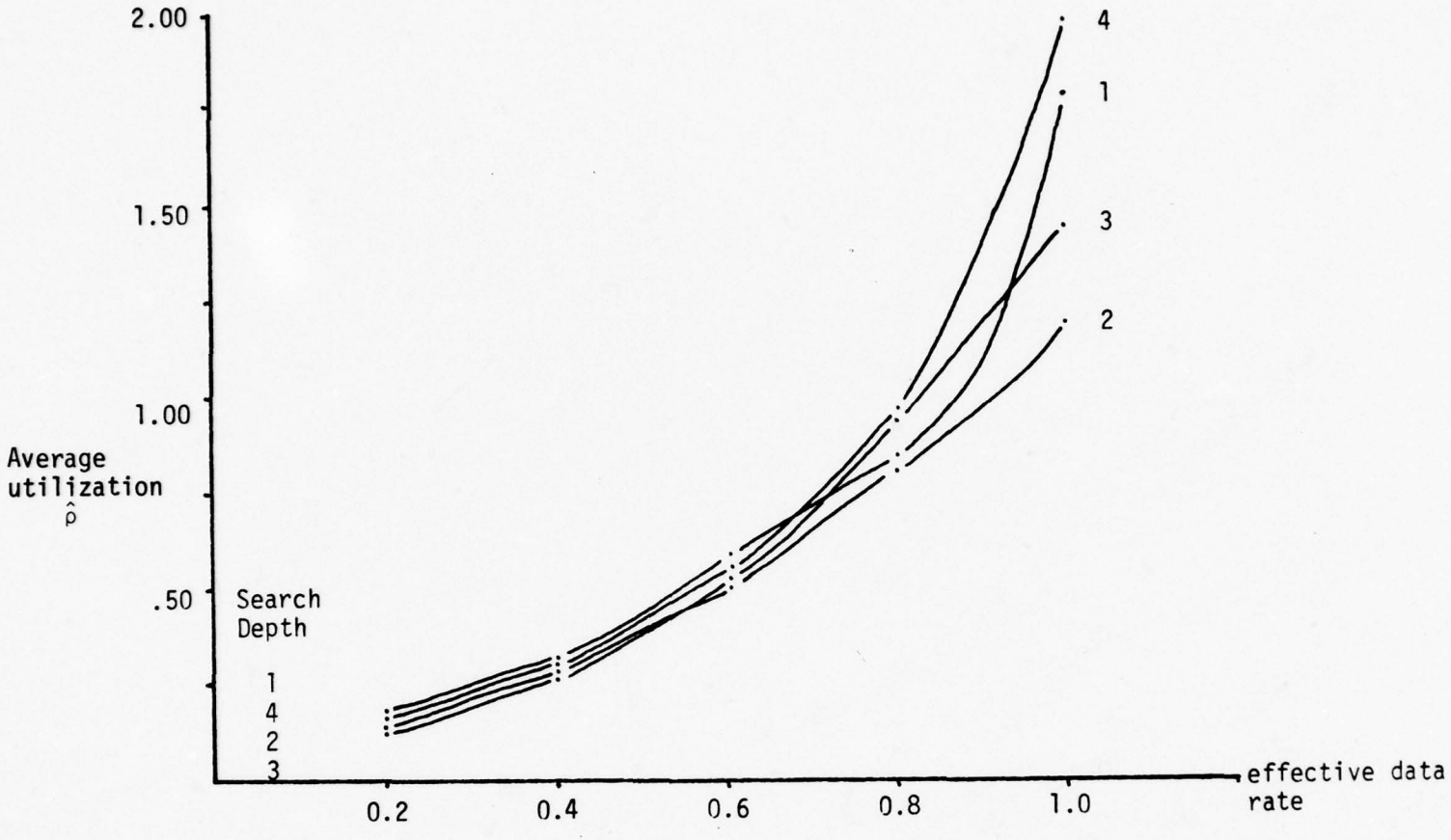
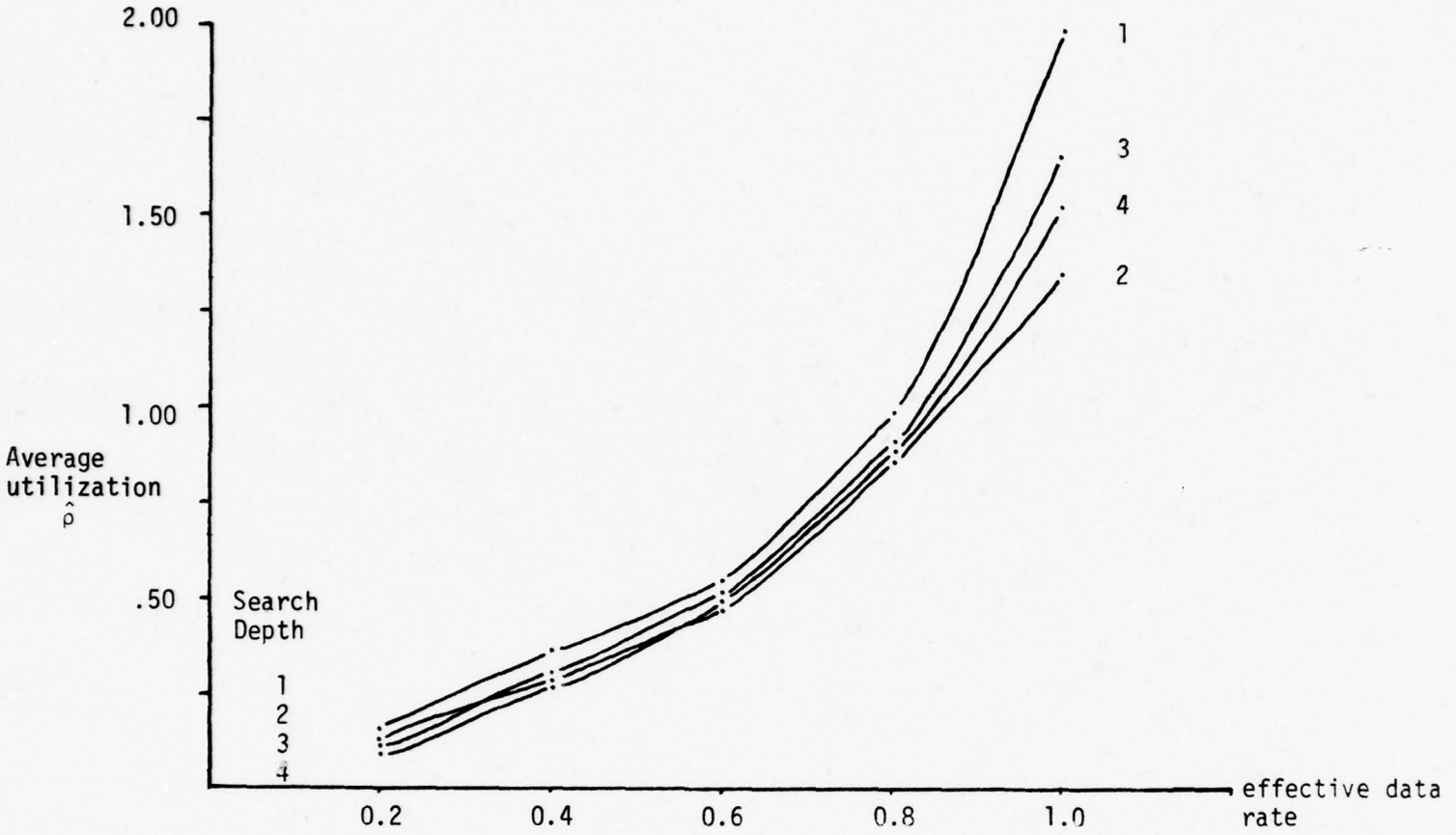


Figure 29. Utilization for 36 node network



lines in these figures are only intended to convey estimates of continuous functions for which adequate mathematical formulae are lacking. Also, the tuning process for each size network was completely independent, making comparisons between performance data from different size networks invalid.

Several independent simulations were made at each of the load factors (RE) 0.2, 0.4, 0.6, 0.8, and 1.0 (see tables in Appendix B). These independent runs involved increasing the levels of the average number of messages delivered per node. Twenty simulations were made at each of the four search depths per network configuration. The results at each load factor were then averaged with the appropriate weight applied to each level of delivered messages. For example, the average packet hop entries, $\hat{j}$, in Table B1-1 were obtained by

$$\hat{j} = \left(\sum_{m=1}^{c} 50mnT_{1,m}\right)/\left(\sum_{m=1}^{c} 50mn\right) \tag{5.1}$$

where n is the number of nodes in the network and T is a matrix representation of Table B1-1. The bounds on the 95 percent confidence intervals were derived using the cumulative t-distribution:

$$L_{.95} = \hat{j} - t_{0.95(c-1)}s/\sqrt{c} \tag{5.2}$$

$$U_{.95} = \hat{j} + t_{0.95(c-1)}s/\sqrt{c} \tag{5.3}$$

where c is the number of columns in Table B1-1, c - 1 is the degrees of freedom, and s is the standard deviation of the entries $T_{1,m}$, $1 \geq m \geq 4$. Confidence intervals for queue lengths were derived using as a weight the number of queue length observations taken during the corresponding

simulations. The respective number of observations is placed adjacent to each average queue length in the queue length tables.

A logical departure point for analyzing search depths is to evaluate the average message hops in traversing a network. Figures 18 through 20 provide representations of average message hops for each of the three evaluated network configurations. Table B1-1 through B1-3 contain the data from which the graphs were derived. Corresponding search depths are indicated for each curve. Numbers directly above or below each data point reflect the maximum hops experienced by any one packet at the indicated effective data rate (EDR). Where more than one number is shown at a given EDR, each number corresponds to increasingly higher search depths. For example, the set {7, 6, 6} describes the maximum hops at search depths 1, 2, and 3 for an EDR of 0.6. The average number of hops per packet, $\hat{j}$, is indicated along the vertical axis and, as with all graphs, the horizontal axis represents the effective data rate (load factor).

Search depths (SD) 1 and 2 both performed well in the smaller, 16 node network. SD 2 was beginning to experience some fluctuations at the extreme high EDR as indicated by a maximum of 12 hops for at least one packet. In contrast, SD 4 maintained a relatively high average hop count through all EDR levels while SD 3 produced fairly irrational results. Both SD 3 and SD 4 indicate strong tendencies towards looping and subsequent network saturation at EDR = 1.0.

The peculiar fluctuations of SD 3 and the minor variation in SD 4 at EDR = 0.6 are contrary to anticipations. They are attributed to the following factors:

1. For a 16 node network, SDs 3 and 4 approach the maximum length of a minimum path across the network. Since destinations are generated randomly only approximately one half of the messages generated for these SDs are ever evaluated for traversing more than two levels. Therefore, relative small variations in average hop count, delay, etc., are greatly amplified when compared to SDs 1 and 2.

2. The relatively short queue length of SD 3 at EDR = 0.2 allows messages to be communicated through a node at a much faster pace. These rapidly changing conditions are believed to be out of phase with timing of delay table updates. As the load increases, maximum hops decrease with a corresponding decrease in average hop count. Note that although queue length remains approximately constant between EDR = 0.4 and 0.6, average delay actually decreases. A properly synchronized delay table update interval is important to the efficient use of network bandwidth.

In the 25 and 36 node networks, SDs 2 and 3 produced similar average hop counts. SD 4 approaches the behavior of SD 1 at the higher EDR in the 25 node network. In the 36 node network there is a tendency toward looping by all four SDs at the higher EDR. SDs 3 and 4 both appear to experience looping at lesser EDRs than does SD 2. Minor variations are attributed to the randomness in which destination addresses are generated.

Clearly, SD 1 performed poorly for the 25 and 36 node networks. The minimum number of hops for the maximum paths across the two networks is 8 and 10 respectively. Maximum hop counts as large as those for SD 1 produce delays an order of magnitude greater than do SDs 2, 3, and 4. It is very apparent that the strong propensity for looping by SD 1 produces the overall higher average hop count.

The correlation between average hop count, average queue length, and average delay is, for all practical purposes, exact. It should be clear that greater hop counts will produce larger queue lengths resulting in more delay (Figures 21 through 26).

One of the more common measures of system utilization is the utilization factor as defined by the ratio of the rate at which "work" enters the system to the maximum rate (capacity) at which the system can perform the work (49). The system is said to be in a state of saturation whenever the arriving work load exceeds the rate at which the work is performed.

The mean utilization factor, $\hat{\rho}$, for the three evaluated networks is illustrated in Figures 27 through 29. Data points in these graphs were obtained by dividing the average arrival time into the average service time. Each data point represents 8,000, 12,500, and 18,000 observations for the 16, 25, and 36 node networks respectively. A network is saturated or approaching saturation whenever $\hat{\rho} = 1$. The utilization factor is one of the better means of combining measurements used to describe a network's operating state.

The relatively poor performance of SDs 3 and 4 on the 16 node network is now re-examined using network utilization as the measurement parameter. The poor performance is believed to be a result of out-of-phase delay at levels greater than two. The period between the time of evaluation and the time that a packet has traversed two more levels is sufficient to allow the previously evaluated minimum path to become loaded. As a result, the network appears as a surface possessing local minima which attract units of work. The greater the search depth, the greater the attraction so that additional minima are created as a result of the

strong attraction. These extremely dynamically fluctuating conditions have a nullifying effect on weight applied to the minimum delay evaluation by the previously described heuristic function. The opposing influence of evaluating levels 3 and 4 explains the looping tendency at higher EDRs in larger networks.

The reason for the poor performance of SD 1 on the larger networks is opposite to that for SDs 3 and 4. Evaluating only one level lacks sufficient intelligence about conditions further down the path. As a result, many packets encounter heavy loads at the second level which cannot be anticipated using SD 1.

In contrast, if a network is sufficiently small that a greater number of packets are generated to nodes within two links of the originator, SD 1 will indicate a better performance. Specifically, if a packet is generated to an adjacent node, the route is immediately fixed. If a packet destined for a node two links away traverses the first link as a minimum cost path, heavy loads down the second link are again overridden because of the adjacency of the packet to the destination. A 16 node network provides such an environment for SD 1 since more than 50 percent of the messages generated will be within a two link distance. The utilization performance on the 16 node network is only slightly worse than the performance of the best search depth. The monotonically increasing curvature of the function connecting its data points signifies little irregularity as increasing loads are encountered. As network size increases, SD 2 excels in performance. The superiority of SD 2 demonstrates a willingness to recover from peak loading, EDR > 1.0, at a much faster pace than the other search

depths. It therefore was selected as the optimum SD and used as a measure to compare other versions of the CAT.

5.2 Evaluating Other Coordinate Address Techniques (CATs)

The evaluation process to determine the best performing SD produced counter-intuitive results. Delays evaluated by various SDs are little more than weighting functions for determining the optimum path. Further adjustments to SD 2 could possibly produce an even better performance, perhaps SD 1.5 or some other fractional search depth. CAT_n, as presented in section 5 was defined with this consideration.

Figures 30 through 32 provide the results of simulating CAT_n on the 36 node network. For economical reasons and because greater variations tend to occur at higher load factors, the simulation was limited to the top three EDRs where greater spreads occur. CAT_1 data were available from previous simulations to evaluate the optimum search depth.

The inverse exponential weighting factor produced the best results among the three evaluated CATs. CAT_2, which used an inverse SD level weight, produced counter-productive results which were even less efficient than SD 1. The correlation of flows in a network prevents the use of analytical analysis for rationalizing variations in each CAT.

The improved results obtained by using CAT_3 are attributed to some relationship between the inverse exponential weight and the Poisson arrival distribution (which produces interarrival times exponentially distributed). The poor performance of CAT_2 is attributed to a weighting function, 1/n, which is diametrically opposed to the cumulative delay of searching n levels. Step 2 in section 5 has been completed.

Figure 30. Average queue lengths for CAT_n

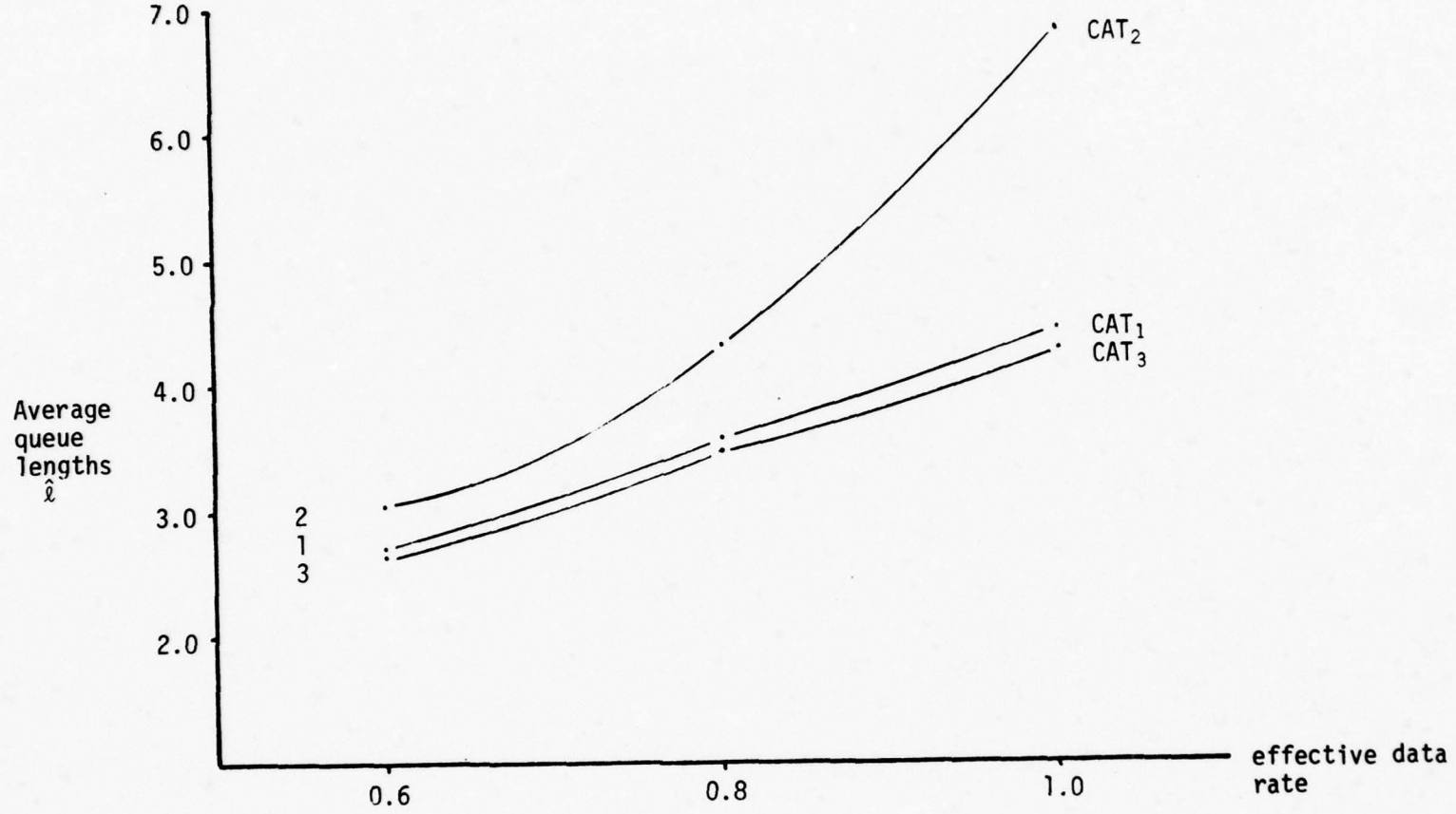


Figure 31. Average delay for CAT_n

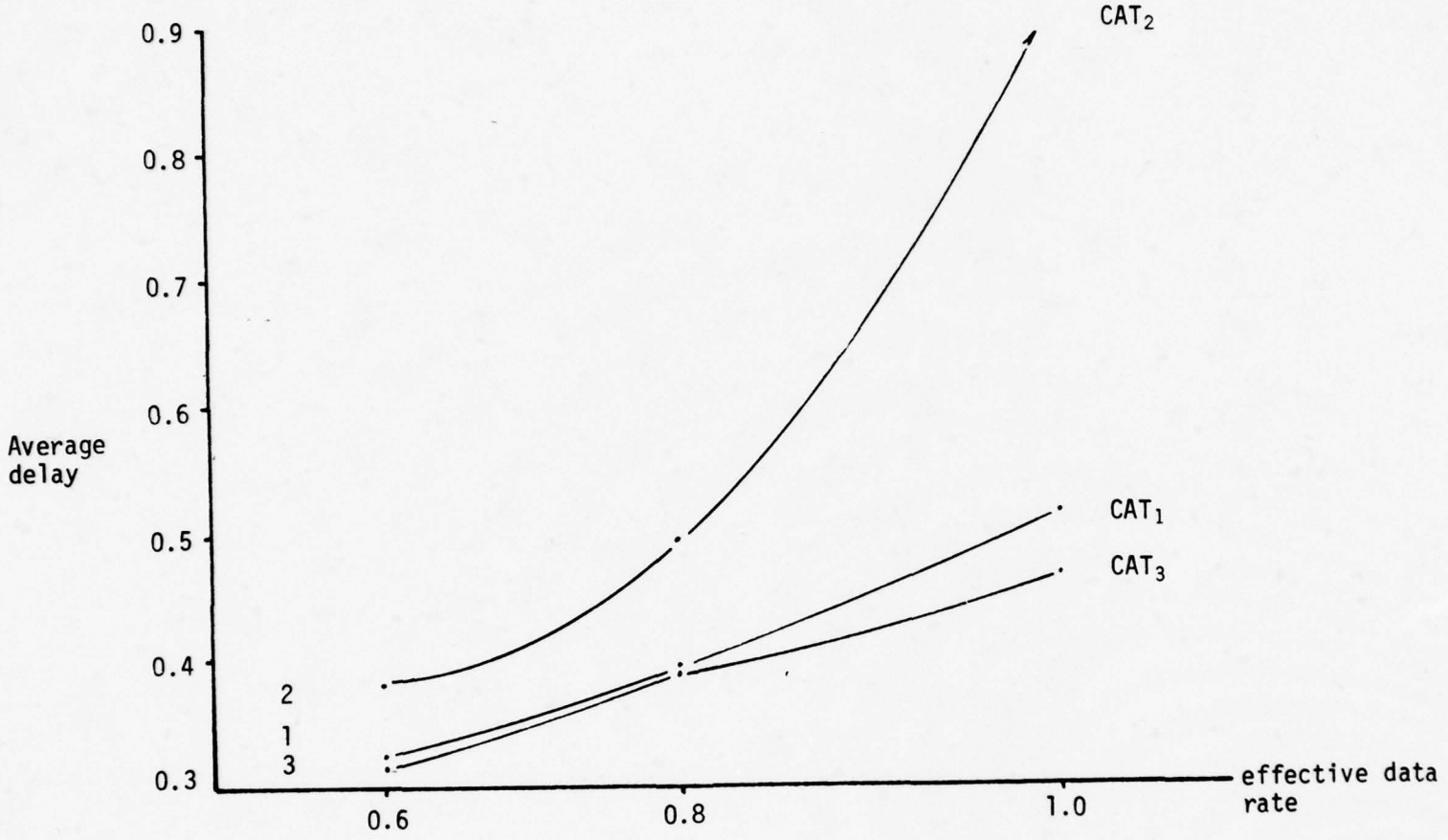
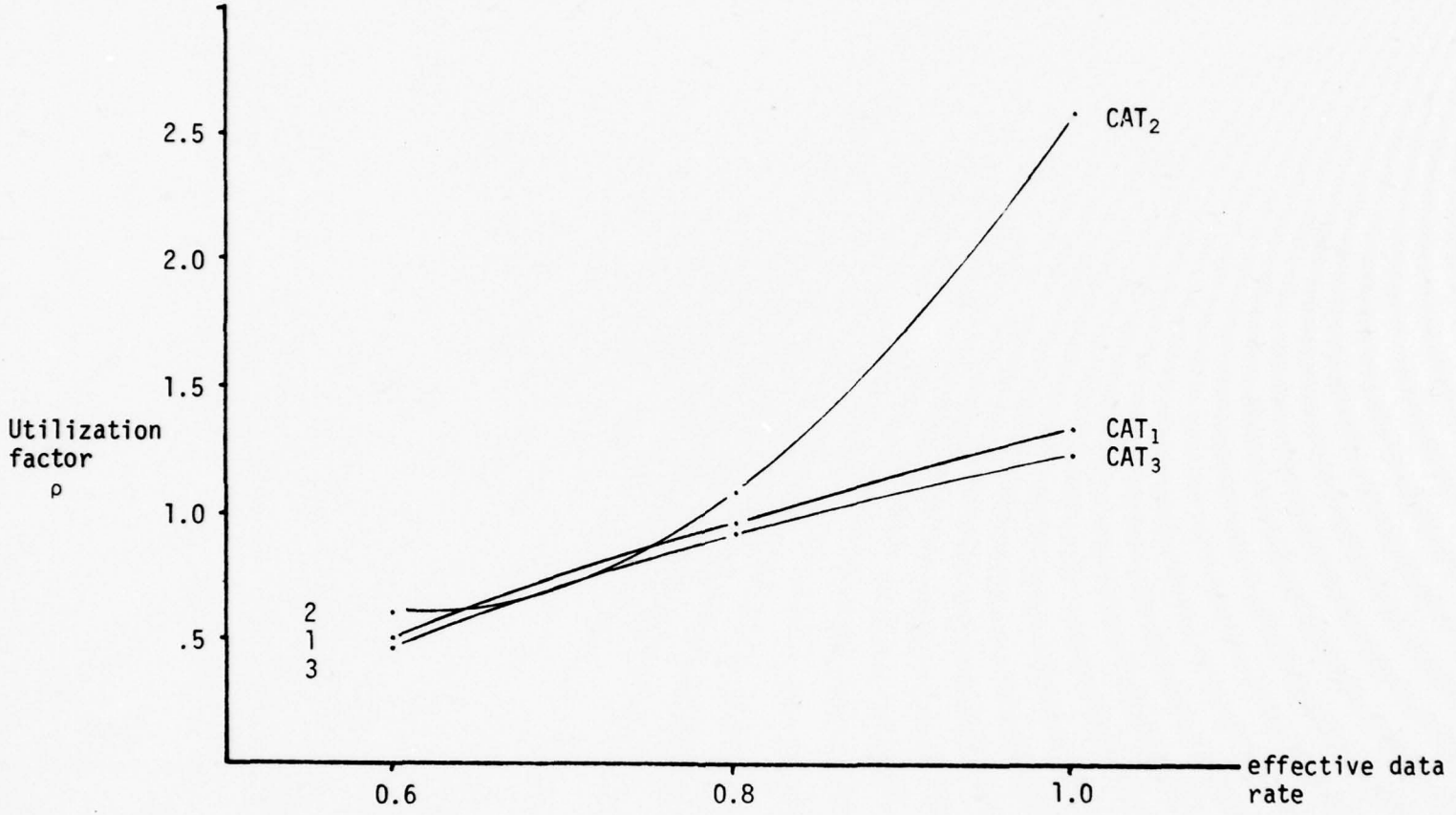


Figure 32. Average utilization for CAT_n



5.3 Establishing Confidence in the Simulator

Law (52) describes a procedure for determining efficient estimators in simulated queuing systems. For example, the efficient estimator for delay is defined as

$$d = \bar{D}_c / \bar{N}_c \tag{5.4}$$

where $\bar{D}_c$ and $\bar{N}_c$ are, respectively, sample means of the total delay of all customers served and the number of customers served in a busy cycle. Similar equations are provided for queue length, waiting time, and others. The concept of determining confidence intervals with the use of efficient estimators has become an accepted practice for infinite or even very large populations.

Estimators may be obtained from the simulations by weighting the means derived from each simulation with the respective number of observations. This procedure was used to calculate the data points for each of the measurements described in this chapter. An example using queue lengths was discussed in section 5.

Steps 1 and 2 of section 5 led to the conclusion that SD 2 refined by CAT_3 is the most optimum of the evaluated algorithms. A complete sequence of simulations was made using this combination on the 36 node network. Means were then derived from the resulting statistics from which confidence intervals were calculated. The 95 percent confidence intervals for queue lengths, delay, and utilization are plotted in Figures 33 through 35.

Solid lines at discrete values along the horizontal axes represent actual confidence intervals calculated for each load factor. Dashed lines are projected approximations for the intervals between discrete

FIgure 33. 95% confidence interval for queue lengths using CAT_3

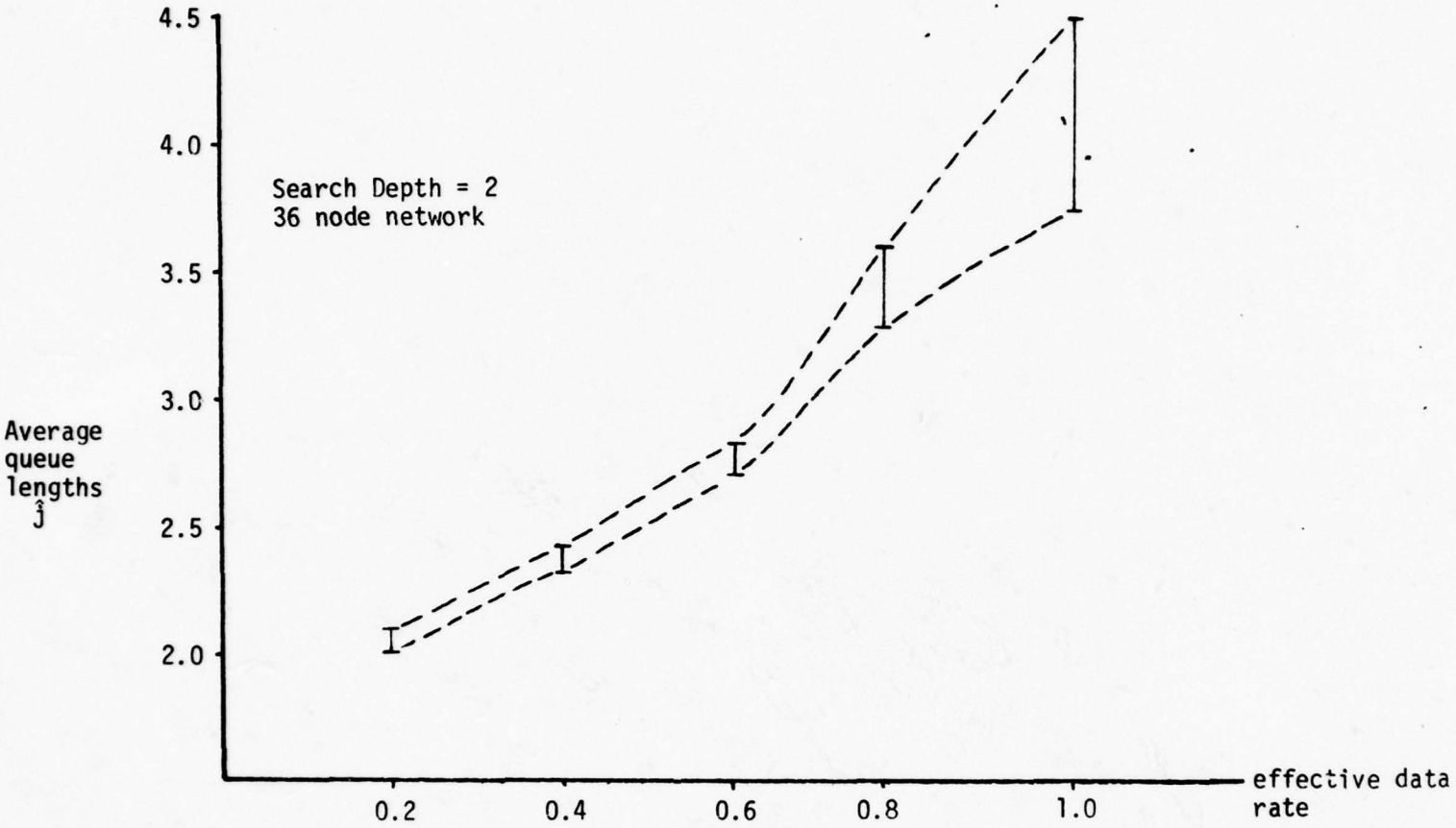


Figure 34. 95% confidence interval for delay using CAT_3

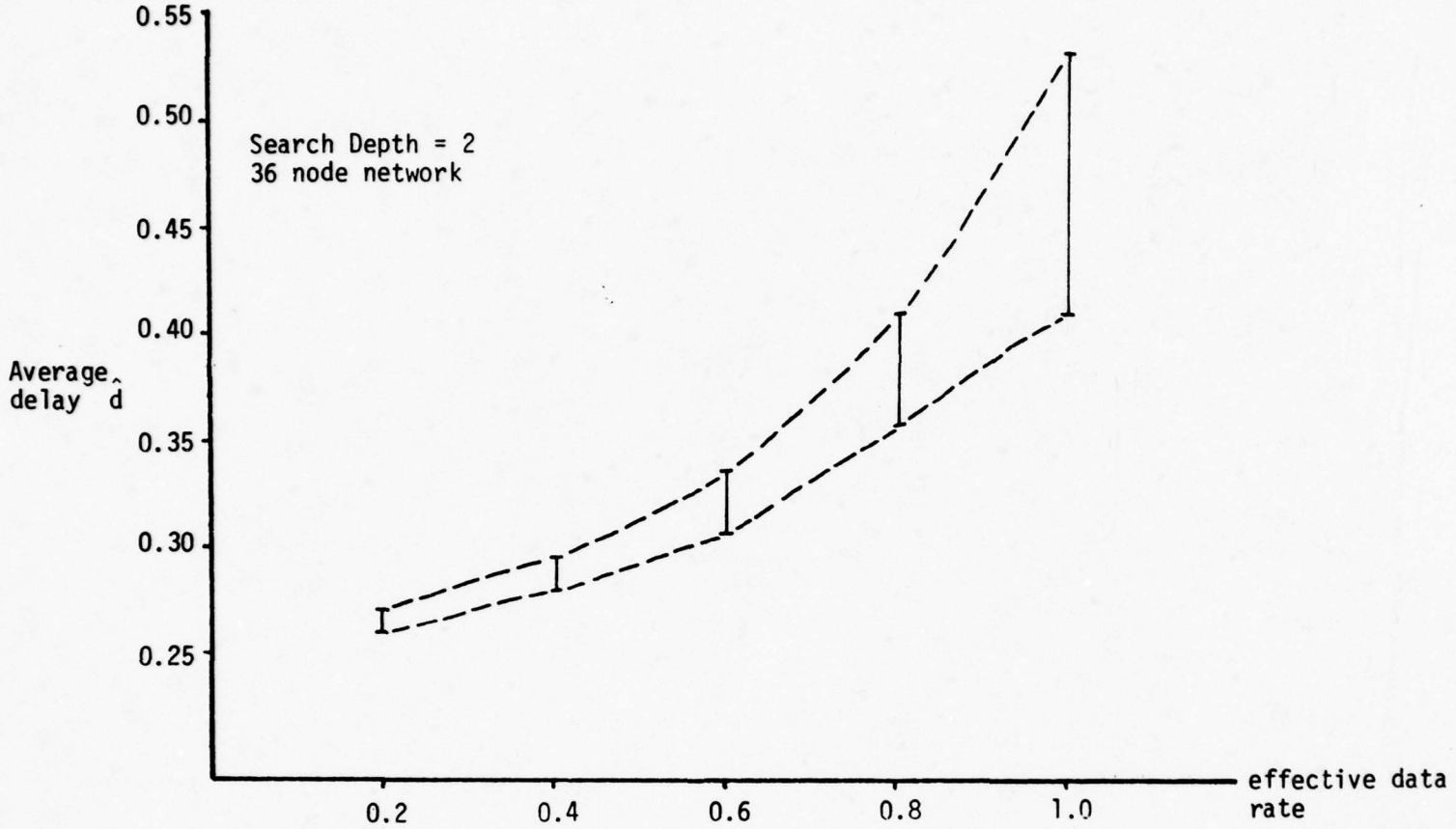
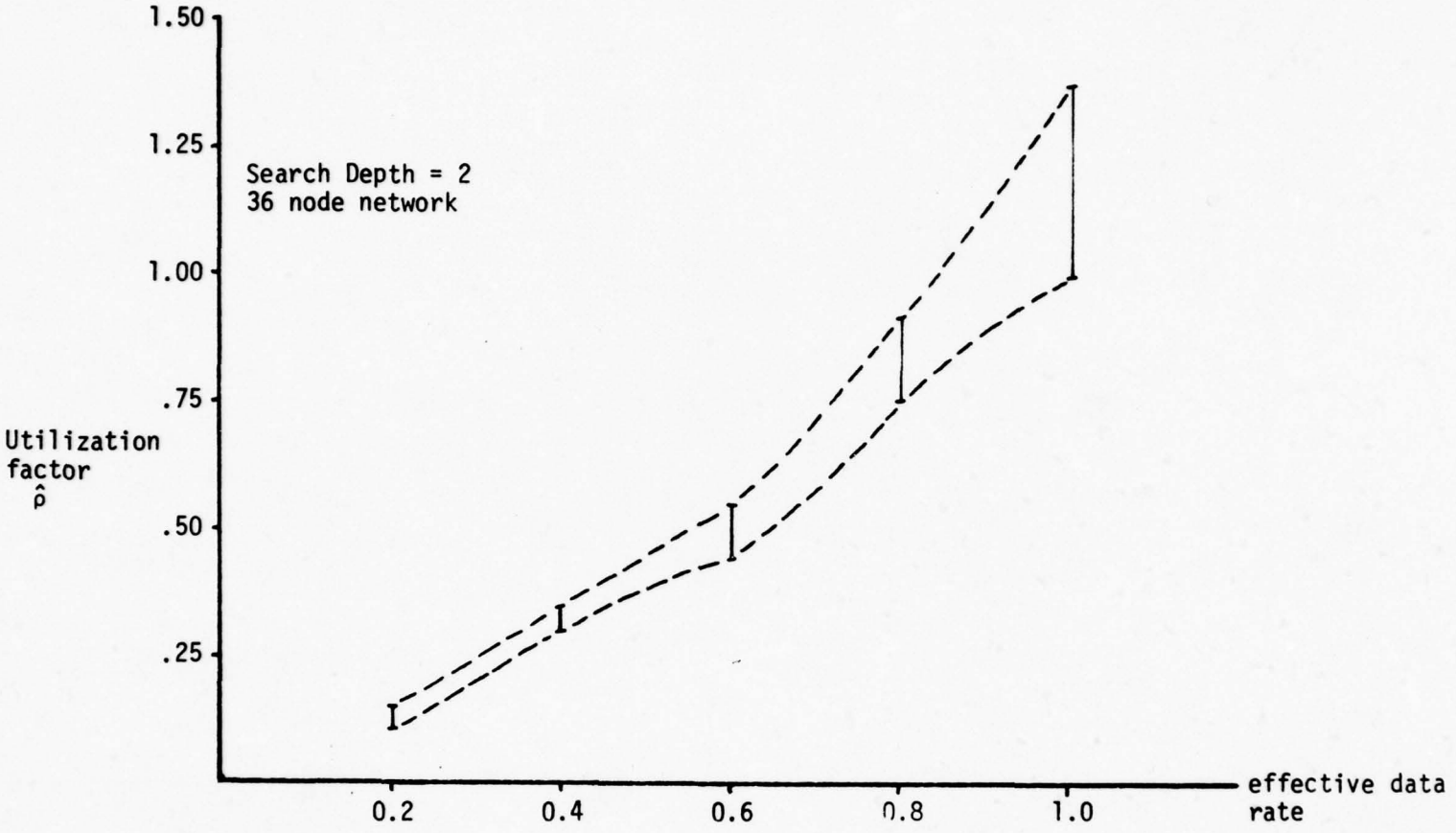


Figure 35. 95% confidence interval for utilization using CAT_3



EDR measurements. Over 640,000 observations were made for queue lengths to establish a 95 percent confidence on the simulated data. Approximately 18,000 observations were made to determine confidence intervals for delay and utilization. The tabulated statistics are contained in Table B1-14.

5.4 Comparative Performance Evaluation

In determining its relative efficiency, CAT_3 was compared to three other algorithms. Two of these, referred to as the periodic update algorithm (PUA) and the global mapping algorithm (GMA) have been thoroughly investigated and applied to several operational networks. The third, called the regional mapping algorithm (RMA), is proposed by Neblock (65) as a solution to looping. Each of these algorithms requires some form of a delay table for preserving delay information. The following discussion provides a functional description of each and how each algorithm compares to the performance of CAT_3.

The PUA, in its basic form, mades no attempt to alleviate the looping phenomena. Each node builds a delay table containing anticipated delays to all other nodes in the network over all outgoing links. The algorithm causes transmission of delay table updating vectors to all nodes after some prespecified interval of time. The communications of update vectors occur approximately synchronously in time for all nodes (33). The PUA operates as a fixed routing procedure until the delay tables are altered. Table III provides a brief description of the algorithm.

Table III. Periodic Update Algorithm

1. If this is the source node, append the source identifier and route the message to its destination via the minimum delay line.

2. If this is the destination node:

   a. remove the message, else

   b. append the current node's address,

   c. select the minimum delay line from the delay table and queue the message for transmission then

   d. go to 2.

Obviously, the PUA will experience looping when delay table updates are no longer responsive to actual conditions. This may occur when loads fluctuate out of phase with table updates, resulting in the GMA which uses a trace field to maintain a historical path of a packet. Although loops cannot occur with the GMA (described in Table IV) messages can become trapped when all adjacent nodes have been visited and the destination has not been reached. Trapping causes many lost messages, defeating a basic philosophical constraint that messages accepted by a network must be delivered. As shown by the following

examples borrowed from Neblock, it is equally damaging to both the GMA and the RMA.

The phenomena of trapping in the GMA is best explained with the use of a diagram (Figure 36). Assume that a packet has been generated at node 1 destined for node 5. Congestion within node 1 forces it to select the port to node 2 for transmission. The packet then follows a path from node 1 to node 2 then to node 4. In the interim, a new minimum delay vector has been issued resulting in delay tables being updated throughout the network. Node 4 subsequently identifies the port to node 3 as the path of minimum delay to node 5. Having previously visited all adjacent nodes, the packet becomes trapped upon being received by node 3.

Figure 36. Message trapping in the GMA

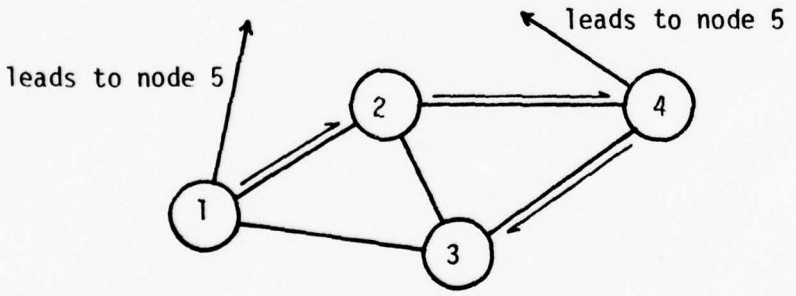


Table IV. Global Mapping Algorithm

1. If this is the source node, turn the source map bit on and route the message via the minimum delay line.	
	2a. If this is the destination node, remove the message, else,
	2b. turn the current node map bit on,
	2c. select the minimum delay line from the delay table; retrieve the node map for the node terminating the minimum delay line; AND the message trace with the next node map; if the result is zero, route the message, else
	2d. select the next best output line from the delay table and return to step 2c.

As a network using the GMA increases in size, available bandwidth for user text decreases. This is because every node in the network requires its own personal bit in every packet. The resulting overhead makes the use of the GMA infeasible for medium and large scale networks. Neblock addresses this problem to some extent in his investigation of loop free algorithms. The RMA, referred to as partitioning, provides a partial solution to the diminishing bandwidth problem for medium scale networks. However, it, too, may become a victim of the trapping

phenomena as explained in the following paragraph. Table V contains a functional description of the RMA.

Figure 37 represents an arbitrary network whose partition 2 possesses connectivity to adjacent partitions as shown. Assume that a packet originated at node 2 and traverses the path 2-4-5-8-7-6-3. It has become trapped at node 3.

Figure 37. Trapping with the RMA

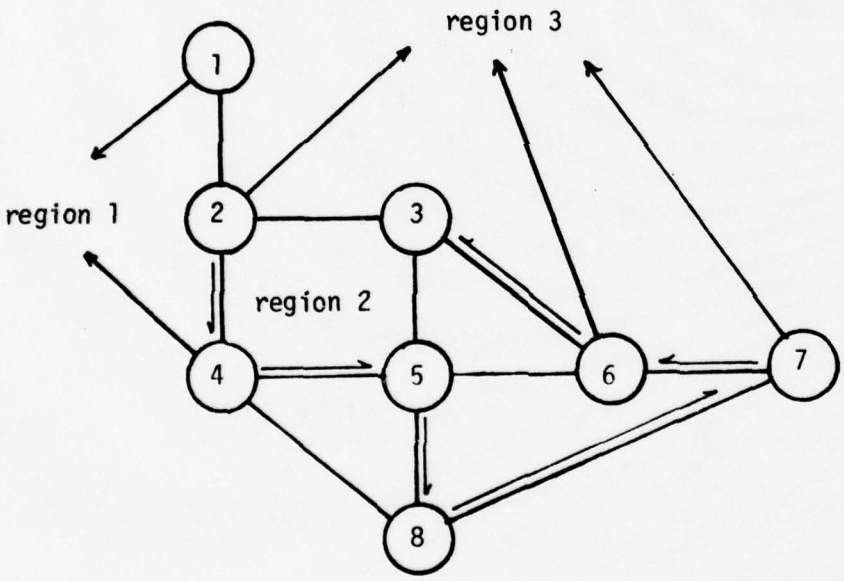


Table V. Regional Mapping Algorithm (65)

1a. If this is the source vertex and it is an interior node, set the region bits to the current area, turn the source map bit on and route the message to the node on the minimum delay path to the destination, else	
1b. if this is the source node, as well as a boundary node bordering a region not previously traversed, route the message to the next region; else route the message to the node on the minimum delay path within the source region.	
	2a. If this is the destination node remove the message, else
	2b. if this is a boundary node to the last region traversed, set the region bits to the current area and turn on the node map bit, else
	2c. if this is a boundary node to the next region to be traversed, route the message to the next region, else
	2d. select the minimum delay line from the delay table, AND the trace map with the next node map; if the next node has not been previously traversed, transmit the message, else
	2e. select the next best output line from the delay table and return to step 2d.

Note that step 2b has the effect of labeling all nodes in the region just exited as previously visited. The advantages and disadvantages of the RMA have been provided earlier. Reference is made to Chapters III and IV for detailed descriptions of CAT_3.

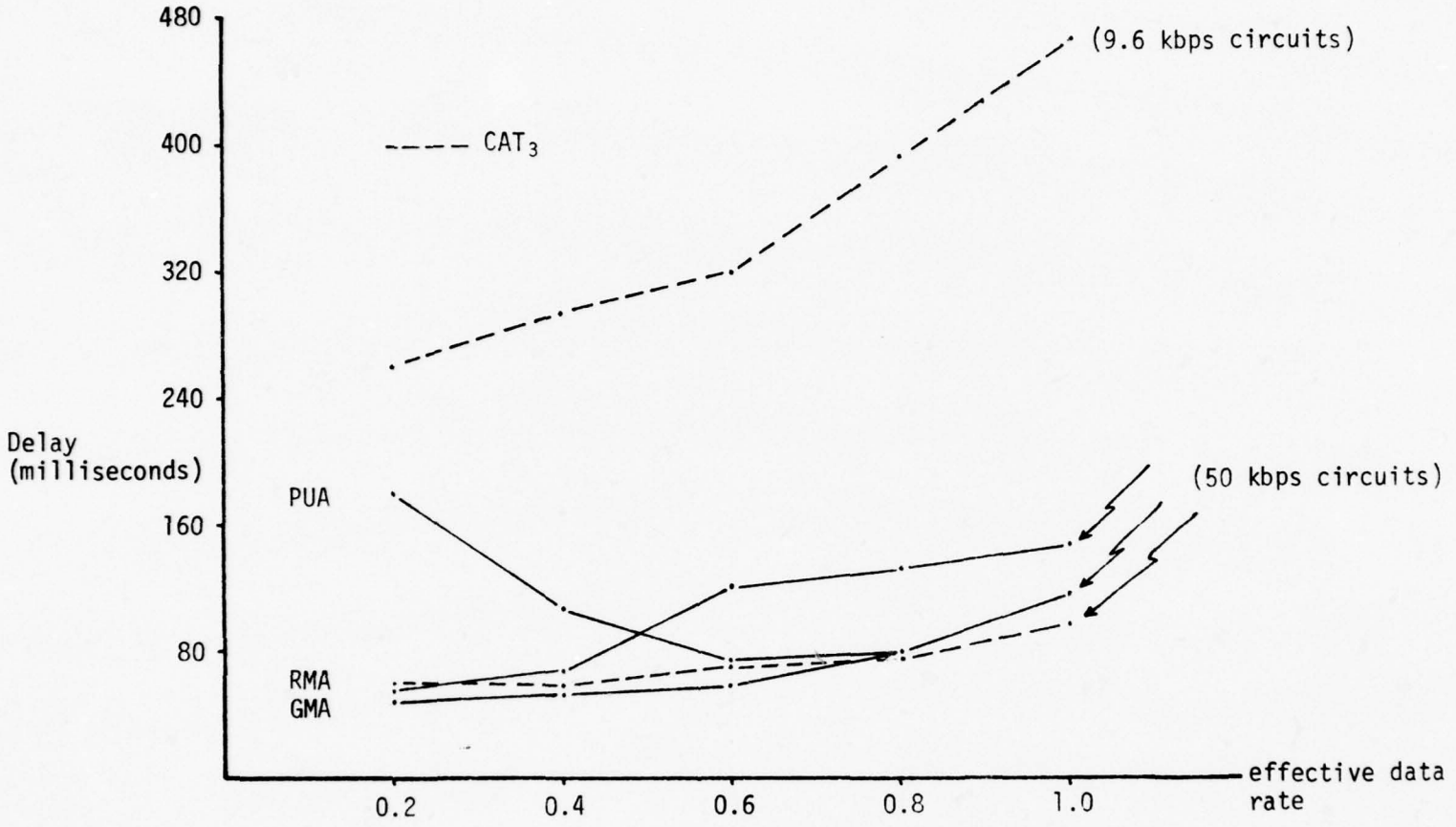
Performance data extracted from reference 65 are used to describe the the PUA, GMA, and RMA. Figure 38 compares the delay performances of the three algorithms to CAT_3. The simulator was reprogrammed for 50 kilobit circuits using a 640 millisecond update interval (as used by Neblock) so that more valid comparisons could be made. The results of using 9.6 kilobit circuits are also plotted.

The solid line curves, representing the simulation of the PUA, GMA, and RMA, were derived using a 19 node network configured similar to an early version of the ARPA network. The 50 kilobit CAT_3 data were obtained using the 25 node distributed network. Assuming no topological irregularities, performance of CAT_3 would tend to be somewhat better on the 19 node ARPA network since the minimum path between maximally separated nodes has only four links versus the eight links for the 25 node network. The similar performance of CAT_3 to the RMA and GMA serves to strengthen the confidence assumed for the simulator.

5.5 Results of Large Scale Network Simulation

Extreme care was taken in tuning the simulator for large networks. Specifically, the simulator was tuned for 256 nodes to provide approximately the same utilization measurements at load factor 0.2 as obtained when simulating the 36 node network. This was necessary for two reasons: 1) to provide for estimates of required (real) resources and 2) to allow projections on the validity of the 256 node data when compared to the

Figure 38. Average delay comparison



36 node data. The results of each simulated load factor, averaging 87 minutes of execution each, were carefully analyzed before proceeding to the next load factor. A final run at load factor 1.5 was made after it was determined that the larger network could accommodate higher load factors before reaching saturation. The following discussion is an aggregate of the analysis on all six load factors. Table B1-16 contains the data that supports this analysis (Figures 39 through 42).

The sharp incline for the average number of hops (Figure 39) between LF = 0.2 and LF = 0.4 is attributed to the fixed routing characteristics of the network under lightly loaded conditions. At load factor 0.2 the average delivery time is computed to be 2.75 seconds (Figure 41) while the average interarrival period is determined to be 20 seconds. Clearly, most messages arriving at any one node are already delivered before succeeding messages arrive. Average queue lengths are relatively short and therefore have little influence on the delay anticipated along a particular path. Each message traverses the network along a fixed route computed by the algorithm (CAT_3) to have the minimum distance between a given originator/destination node pair.

The routing characteristics of CAT_3 tend initially to force messages which must traverse greater distances toward fixed routing. This is because of the overriding influence of the distance factor when combined with delays contained in the delay table. As a message gets closer to its destination, the relative influence of the two factors reverse with delay table entries becoming more dominant. This causes the routing of messages with short distances to be more responsive to fluctuating load

Figure 39. Average hops for 256 node network

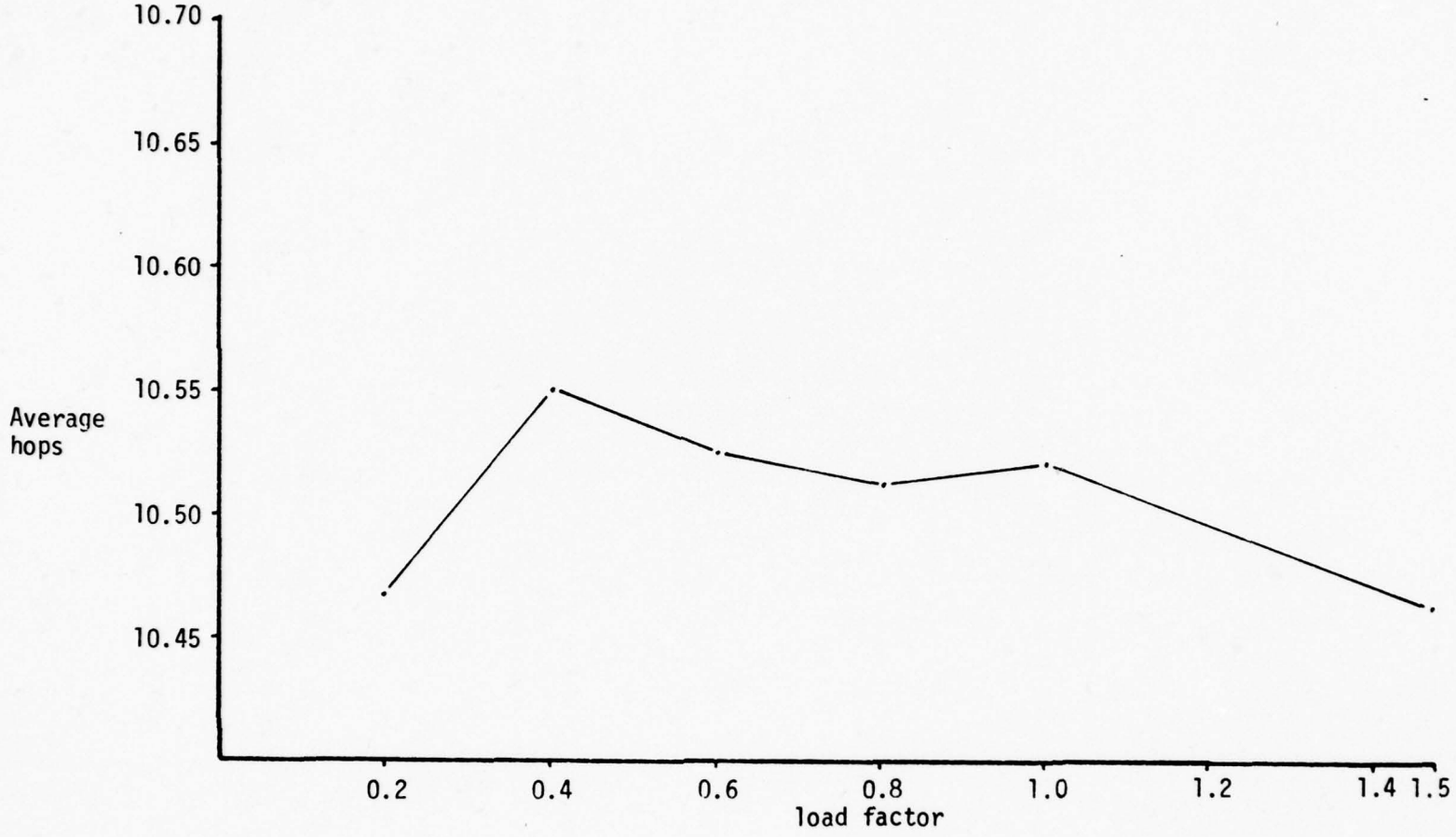


Figure 40. Average queue length for 256 node network

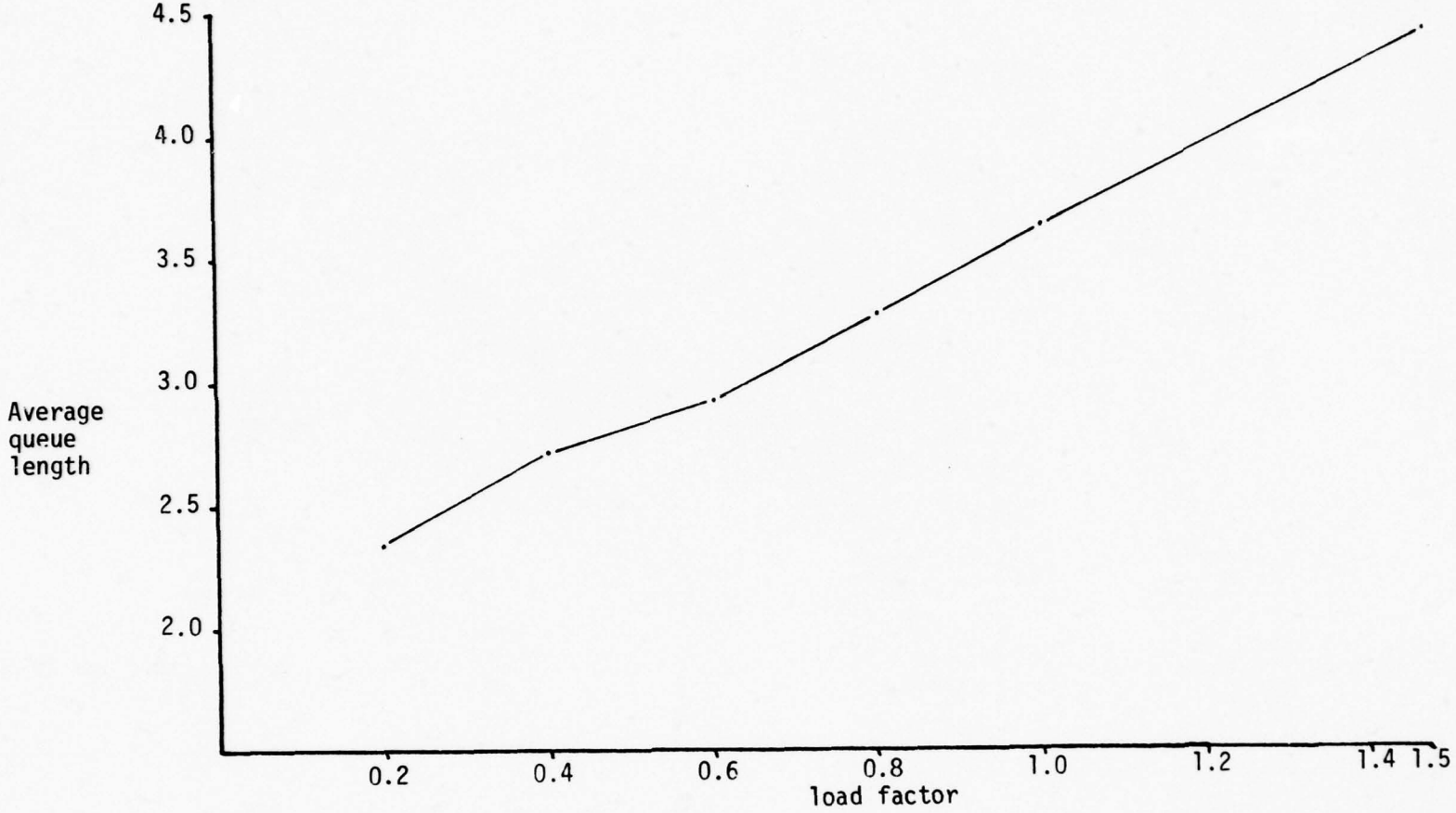


Figure 41. Average delay for 256 node network

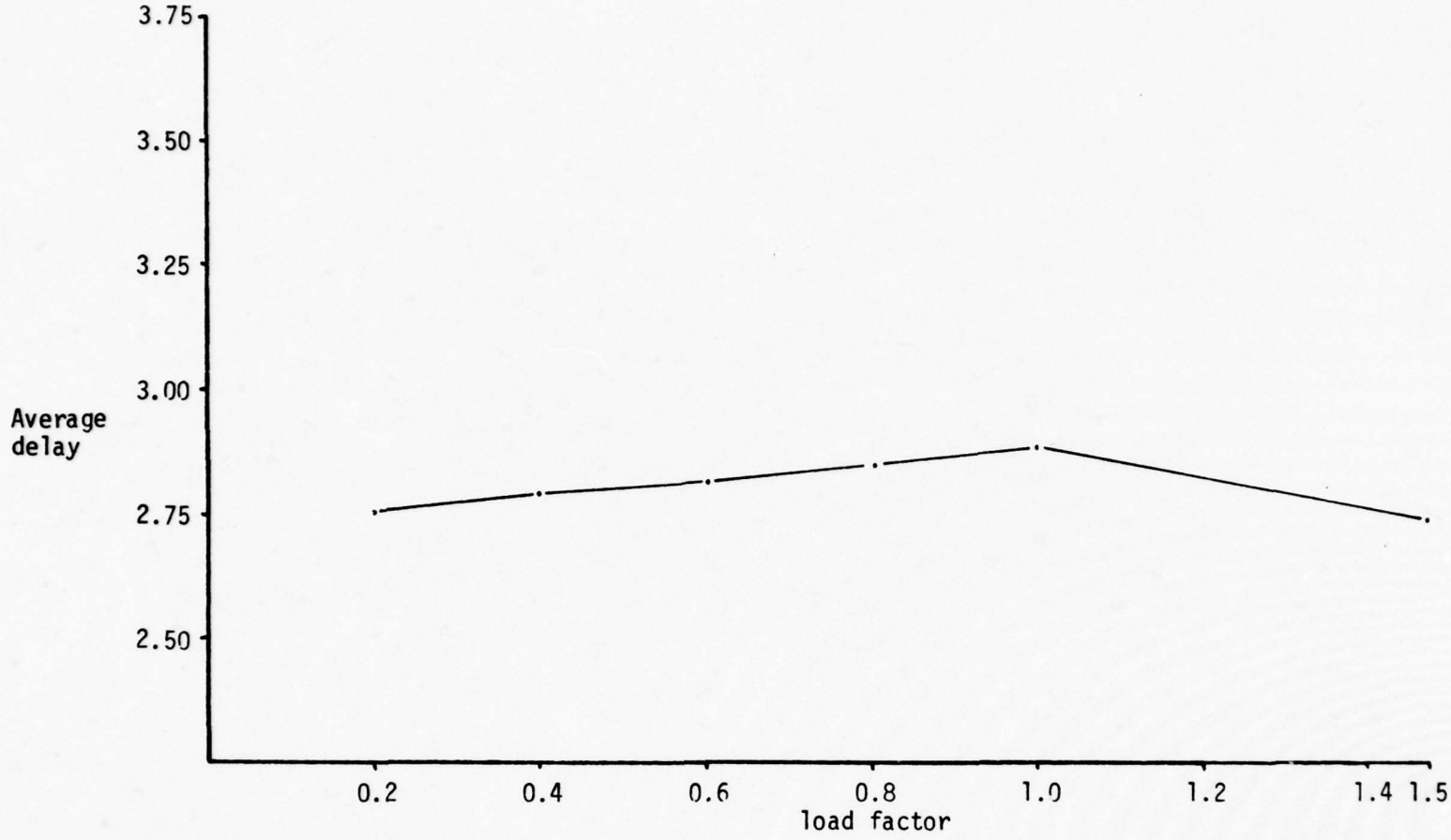
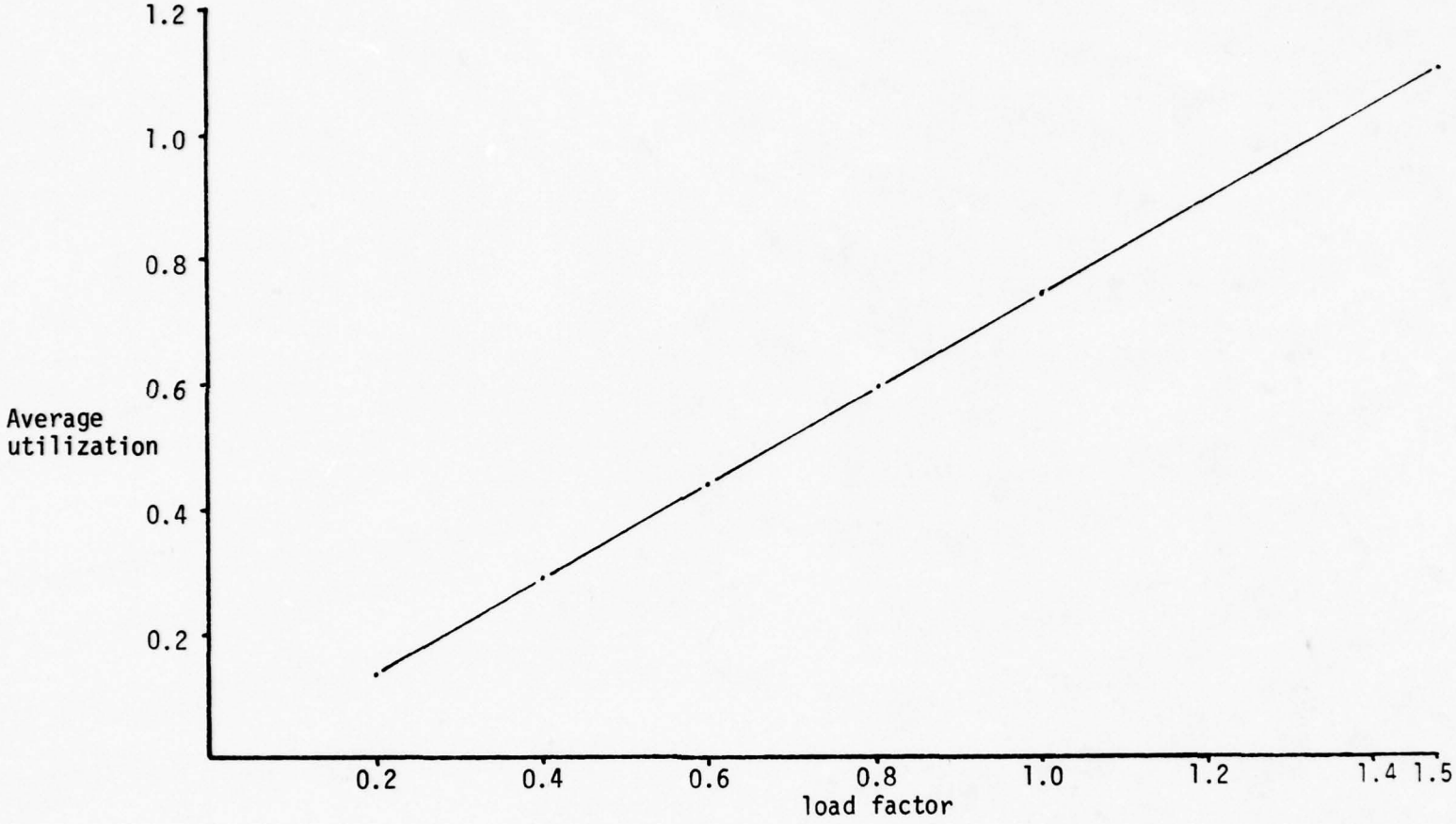


Figure 42. Average utilization for 256 node network



conditions. The conclusion is that as messages get closer to their destinations, a greater emphasis is placed on "time to delivery" and less on "distance to travel".

The analysis indicates a trend of decreasing average hops per message for load factors greater than 0.4. Average queue lengths continue to increase at a linear rate. The same is true for utilization which is derived as the ratio of average delay to average interarrival time. The average delay remains relatively constant with minor increases at each higher load factor.

The length of a simulation is a direct function of the total number of messages delivered, i.e., the run is terminated when the average messages delivered per node reaches a prespecified level. Therefore, as load factors increase, a greater percentage of messages delivered require fewer hops than at lower load factors. Messages are arriving at a faster pace and those with shorter distances to travel are being delivered at a higher rate.

As anticipated, average queue lengths increase at a linear rate with increased load factors. A linear increase is expected until looping occurs at which time the network saturates. Queue lengths then grow at an explosive rate with corresponding growths in average hops, delay, and utilization.

Based on analysis of data on the 36 node network and the rate of utilization increase resulting from increased load factors on the 256 node network, saturation would have occurred at approximately LF = 2.0 where utilization is expected to exceed 1.3. In contrast, a 36 node network will eventually saturate if it is allowed to operate at LF = 1.0

for extended periods. The difference is attributed to the greater number of routing alternatives in large networks. To emphasize, the dynamic aspect of the routing algorithm affords larger networks with a greater distribution of fluctuating loads. One concludes that for a given utilization starting value, larger networks maintain stability much longer than smaller networks under increasing load conditions. In that respect, CAT_3 has demonstrated its robustness to varying loads in large networks.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

6. Conclusions

The primary objective of this research was to investigate and develop a routing algorithm feasible for large scale networks. The algorithm must retain as many of the desirable properties of current adaptive routing policies as possible. Utilization, as defined by classical queuing theory, is presented as the basic efficiency metric, although several other performance measurements were made.

Initial investigations were devoted to minimizing the memory impact for maintaining delay tables. For modeling purposes, a network is described as a set of trees so that delay information at each node need only be maintained to the farthest leaf in the tree. The size of the tree was evaluated for optimum depth. This portion of the research proved to be beyond intuitive expectations since, contrary to that perceived, a search depth of two was concluded to be optimum. Not only is the execution overhead for selecting a minimum path significantly decreased over a search depth of 3 or 4, but the table size for delay data is greatly reduced by a search depth of two.

Next, it was necessary to develop means for evaluating delay between the leaves of the tree and the destination. A heuristic measure was applied because actual delay is unknown beyond the leaves. The only environmental descriptor of any given destination available to an originator was determined to be a function of the address field and the physical coordinates identifying the destination. As a result,

network topology was configured for use with the rectangular coordinate system. The distance measure applied was the sum of the absolute differences between the respective vertical and horizontal components of the plane.

Various weights were applied to evaluate a search tree, having determined an optimum tree depth and supporting heuristic. This was done because fine tuning search depth two produced an even better performance. An inverted exponential bias proved to be the best of the weights evaluated. Final analysis indicated that consistently good performance was obtained with a search depth of two biased with an inverted exponential weight and increased by a heuristic representation of the remaining delay. The resulting algorithm was the third variation of the coordinate addressing techniques considered and therefore named CAT_3.

Although the algorithms evaluated create noteworthy savings in nodal storage and link bandwidth, looping remains a potential problem. The few loops produced by search depth two occurred at very high load factors, normally above day-to-day operating loads. However, momentary peaks of high loading will tend to produce some loops with the additional delay resulting in dissatisfied customers. The "detect and suppress" methods employed in this research to decrease looping do not prevent loops. A perfect loop prevention mechanism, if one exists for the CAT, has not been found.

6.1 Recommendations

A considerable amount of resources was necessary to collect simulated statistics for this research. Economic considerations prohibit more than a demonstration that CAT_3 will perform well on very large networks. The costs of an in-depth performance evaluation on large scale networks is astronomical in terms of execution overhead.

Further research is necessary to devise methods for better control on conditions which tend to produce looping. More effective techniques should be developed that would allow an algorithm using a heuristic measure of delay to perform without loops. As a practical tradeoff, perhaps there exists a technique between the partitioning scheme and the coordinate addressing technique which may provide a solution to the looping phenomenon.

Finally, a unified theory for large scale network design is necessary which incorporates the experience gained from previous research. Such a project could:

1. Investigate and define parameters that characterize workload and performance of large networks.

2. Develop a standard theory, symbolism, or language to describe the various events and functions in large scale network.

3. Develop a theory of design for large scale networks.

The evolution of large data networks offers many unusual and challenging problems as an advancing technology.

REFERENCES

1. Adkins, G. and Pooch, U.W. Computer simulation: A tutorial. ACM Computer 10, 4 (April 1977), 12-17.

2. Agnew, C.E. On quadratic adaptive routing algorithms. Comm ACM 19, 1 (Jan. 1976), 118-122.

3. Allery, G.D. Data communications and public networks. Proc. of the IFIPS (1974), 114-122.

4. Anderson, G.E. The use of microcomputers in DCS AUTODIN tributaries. Rept. AD-A035-708, NTIS, Dec. 1976.

5. Aupperle, E.M. The MERIT network re-examined. Rept. MCN-0273-TP13, MERIT Computer Network, University of Michigan, Feb. 1973.

6. Baran, P. On distributed communications: An introduction to distributed communications networks. Rept. RM-3420-PR, RAND Corp., Santa Monica, CA, Aug. 1964.

7. Belford, G.G., Bunch, S.F., Day, J.D., et al. A state-of-the-art report on network data management and related technology. Rept. AD-A014-233, NTIS, April 1975.

8. Bina, P.A. Design aspects of a circuit switching system for a national network. Proc. Int. Conf. on Comm. (June 1971), 23-12 - 23-17.

9. Boehm, B.W. and Mobley, R.L. Adaptive routing techniques for distributed communications systems. IEEE Trans. on Comm. Techn. Com-17, 3 (June 1969), 340-349.

10. Boehm, S.P. and Baran, P. On distributed communications: II digital simulation of hot-potato routing in a broadband distributed communications network. Memo RM-3103-PR, RAND Corp., Santa Monica, CA, Aug. 1964.

11. Boorstyn, R.R. and Frank, H. Large-scale network topological optimization. IEEE Trans. on Comm. Com-25, 1 (Jan. 1977), 29-47.

12. Boorstyn, R.R. and Schwartz, W. Computer-communications networks: An introduction. Seminar notes taken from lectures given at Imperial College of Science and Technology, London, England, The Mitre Institute, April 1973.

13. Burke, P.J. The output of a queueing system. Operations Research 4 (1956), 699-704.

14. Chou, W. Planning and design of data communications networks. Proc. of AFIPS NCC 43 (1974), 553-560.

15. Chu, W.W. and Konkeim, A.G. On the analysis and modeling of a class of computer communication systems. IEEE Trans. on Comm. Com-20, 3 (June 1972), 645-660.

16. Clark, S., Krikorian, A. and Ransen, J. Computing the n best loopless paths in a network. J. SIAM 11, 4 (Dec. 1973), 1096-1102.

17. Cole, G.D. Computer network measurements - techniques and experiments. Rept. AD-739-344, NTIS, Oct. 1971.

18. Dantzig, G. Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963.

19. Da Rin Mills, L.A. Queues and network computers. Rept. UIUCDCS-R-73-577, Dept. of Computer Science, University of Urbana - Champaign, Urbana, IL, May 1973.

20. Davies, D.W. The control of congestion in packet switching networks. Proc. of the Second ACM/IEEE Sym. on the Optimization of Data Comm. Sys., Palo Alto, CA (Oct. 1971).

21. Davies, D.W. and Barber, D.L.A. Communications Networks for Computers, John Wiley and Sons, London, England, 1973.

22. Deo, N. Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Englewood Cliffs, NJ, 1974.

23. Doll, D.R. Efficient allocation of resources in centralized computer communications. Rept. AD-862-821, NTIS, Nov. 1969.

24. Fano, R. and Corbato, F.J. Time-sharing on computers, Sci. America 215, 3 (Sept. 1966), 129-140.

25. Farber, D.J. and Heinrich, F.R. The structure of a distributed computer system - the file system. Proc. of Int. Conf. on Computer Comm. (Oct. 1972), 364-370.

26. Fisher, C.R. Introduction to the DATRAN switched digital network. Proc. of the Int. Conf. on Comm. (June 1971), 23-1 - 23-3.

27. Fishman, G.S. Statistical analysis for queueing simulations. Management Science 20, 3 (Nov. 1973), 363-369.

28. Flato, L. Know your common carriers. Computer Decisions (Dec. 1974), 17-30.

29. Frank, H. and Chou, W. Topological optimization of computer networks. Proc. of IEEE (Nov. 1972)

30. Frank, H. and Frisch, I.T. Planning computer-communications networks. Computer-Communications Networks, N. Abramsom and F. Kuo, eds., Prentice-Hall, Englewood Cliffs, NJ, 1973.

31. Frank, H., Kahn, R.E., and Kleinrock, L. Computer communication network design - experience with theory and practice. Networks 2 (1972), 135-166.

32. Fultz, G.L. and Kleinrock, L. Adaptive routing techniques for store-and-forward computer communication networks. Proc. of the 1971 IEEE Int. Conf. on Comm. (June 1971), 39-1 - 39-8.

33. Fultz, G.L. Adaptive routing techniques for message switching computer-communications networks, Ph.d. Dissertation, University of California, Los Angeles, CA, June 1972, 256-257.

34. Gains, E.V., Jr. and Toplin, J.M. The emergence of national networks. Telecommunications (Dec. 1971).

35. Gerla, M. The design of store-and-forward (S/F) networks for computer communications. Rept. Ad-758-204, NTIS, Jan. 1973.

36. Greene, W.H. and Pooch, U.W. A review of classification schemes for computer communications networks. IEEE Computer 10, 11 (Nov. 1977), 12-21.

37. Gross, D. and Harris, C.M. Fundamentals of Queueing Theory, John Wiley and Sons, New York, NY, 1974, 55-56.

38. Hasegawa, Hideo, Miyahara, Teshigawara, Tushihara, and Yoshimi. A comparative evaluation of switching methods in computer communications networks. ICC-75, San Francisco, CA (June 16-18, 1975), 6-6 - 6-10.

39. Hayes, J.F. Performance models of an experimental computer communications network. Bell System Technical Journal 53, 2 (Feb. 1974), 225-257.

40. Hofwell, L. Management planning in the data communications environment. Proc. of the AFIPS Nat. Computer Conf. 43 (1974), 561-565.

41. Jackson, J.R. Networks of waiting lines. Operations Research 5, (1957), 518-521.

42. Kamoun, F. Hierarchical routing procedures for large computer networks. Ph.d. Dissertation, University of California, Los Angeles, CA, June 1976.

43. Kimbleton, S.F. and Schneider, M.G. Computer communication networks: Approaches, objectives, and performance considerations. ACM Computing Surveys 7, 3 (Sept. 1975), 129-179.

44. Kahn, R.E. and Crowther, W.R. A study of the ARPA computer network design and performance. Rept. No. 2161, Bolt, Beranek, and Newman, Inc., Cambridge, MA, Aug. 1971.

45. Kleinrock, L. Communications Nets: Stochastic Message Flow and Delay. McGraw-Hill, New York, NY, 1964.

46. Kleinrock, L. Models for computer networks. Proc. of the Int. Comm. Conf. (June 1969), 21-9 - 21-16.

47. Kleinrock, L. Analytical and simulation methods in computer network design. Proc. AFIPS SJCC (1970), 569-579.

48. Kleinrock, L. Scheduling, queueing and delays in time-shared systems and computer networks. Computer-Communications Networks, N. Abramson and F. Kuo, eds., Prentice-Hall, Englewood Cliffs, NJ, 1973, 95-141.

49. Kleinrock, L. Queueing Systems Volume I: Theory. John Wiley and Sons, New York, NY, 1975.

50. Kleinrock, L. Advanced teleprocessing systems. Rept. AD-A034-111, NTIS, June 1976.

51. Kleinrock, L. and Naylor, W.E. On measured behavior of the ARPA network. Proc. AFIPS Nat. Computer Conf. (May 1974), 299-312.

52. Law, A.M. Efficient estimators for simulated queueing systems. Management Science 22, 4 (Sept. 1975), 30-41.

53. Lindberg, D.A. Automated data acquisition applied to monitoring of communications systems. Rept. AD-A031-474, NTIS, Oct. 1976.

54. Luther, W.J. The conceptual basis of CYBERNET. Computer Networks, K. Rustin, ed., Prentice-Hall, Englewood Cliffs, NJ, 1972.

55. Marchand, N. Priority pricing. Management Science 20, 7 (March 1974), 1131-1140.

56. Martin, J. Teleprocessing Network Organization, Prentice-Hall, Englewood Cliffs, NJ, 1970.

57. Martin, J. Systems Analysis for Data Transmission, Prentice-Hall, Englewood Cliffs, NJ, 1972.

58. McKinney, J.M. A survey of analytical time-sharing models. ACM Computing Surveys 1, 2 (June 1969), 105-116.

59. McQuillan, J. Adaptive routing algorithms for distributed computer networks. Rept. AD-781-467, NTIS, May 1974.

60. Meister, B., Mueller, H.R., and Rudin, H.R. On the optimization of message-switching networks. IEEE Trans. on Comm. Com-20, 1 (Feb. 1972), 8-14.

61. Metcalfe, R. Strategies for interprocess communication in distributed computing systems. Computing and Communications Network and Teletraffic, J. Fox, ed., Polytechnic Press, Brooklyn, NY, 1972, 519-525.

62. Metcalfe, R.M. Packet communications, Rept. AD-771-430, NTIS, Dec. 1973.

63. Millar, J.Z. DCS AUTODIN trunking transmissing between switching centers. Proc. of Int. Workshop on Networks of Computers NDC-68, NSA, Ft. Meade, MD, (Oct. 1968), 221-224.

64. Morgan, D.E., Banks, W., Goodspeed, D.P., and Kolanko, R. A computer network monitoring system. IEEE Trans. on Software Engineering SE-2, 3 (Sept. 1975), 299-311.

65. Neblock, C.E. Loop-free adaptive routing procedures for distributed computer-communications networks, Ph.d. Dissertation, Texas A&M University, College Station, TX, Aug. 1977.

66. Nilsson, N.J. Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, Inc., New York, NY, 1971.

67. Paoleth, L.M. AUTODIN. Computer Communications Networks, Noordhoff International Publication, Leyden, The Netherlands, 1975, 345-372.

68. Pritzker, A.A.B. The GASP-IV Simulation Language, Wiley-Interscience, New York, NY, 1974.

69. Prosser, R.T. Routing procedures in communications networks - part I: Random procedures.. IRE Trans. on Comm. Systems (Dec. 1962), 322-329.

70. Prosser, R.T. Routing procedures in communications networks - part II: Directory procedures. IRE Trans. on Comm. Systems (Dec. 1962), 329-335.

71. Roberts, L.G. Multiple computer networks and mini-computer communications. ACM Sym. on Operating System Principles, Oct. 1967.

72. Roberts, L.G. and Wessler, B.D. Computer network development to achieve resource sharing. AFIPS Conf. Proc. SJCC 36, Atlantic City, NJ, (1970) , 543-549.

73. Rudin, H. On routing and "delta routing": A taxonomy of techniques for packet-switched networks. Rept. RZ-701, IBM Research, Zurich, Switzerland, June 1975.

74. Rudin, H. A performance comparison of routing techniques for packet-switched networks. Rept. RZ-702, IBM Research, Zurich, Switzerland, June 1975.

75. The MERIT Computer Project. The MERIT computer network, progress report for the period July 1969 - March 1971, Publication OJ71-PR-4, University of Michigan, Ann Arbor, MI, Feb. 1973.

76. Tymes, L. TYMNET - a terminal-oriented communications network. Proc. AFIPS 1971 SJCC 28, AFIPS Press, Montvale, NJ, (1971) 211-216.

77. United States Ninety-third Congress. Public Law 93-579, S. 3414, Washington, DC, Dec. 1974.

78. X3S3.4. American National Standard for Advanced Data Communications Control Procedures, Final Draft, ANSI, Washington, DC, Nov. 1976.

79. Yaged, B., Jr. Minimum cost routing for static network models, Networks 1, John Wiley and Sons, New York, NY, (1971), 139-172.

APPENDIX A

The following is a mathematical description of the Markovian birth-death process for the M/M/C system. Standard Kendall notation is used for arrival distribution/service distribution/servers where M implies Poisson and exponential distributions for arrivals and service respectively.

Let $p_n(t)$ = Pr{n in the system at time t} and h represent some small increment in time. Also let the arrival rate be $\lambda = \lambda n \forall n$ and the service rate be

$$\mu n = \begin{cases} n\mu & (1 \leq n < c) \\ c\mu & (n \geq c). \end{cases} \tag{A1-1}$$

Then the differential-difference equations are:

$$p_n(t + h) = p_n(t)(1 - \lambda_n h)(1 - \mu_n h) + p_{n-1}(\lambda_{n-1} h)(1 - \mu_{n-1} h) + p_{n+1}(1 - \lambda_{n+1} h)(\mu_{n+1} h) \tag{A1-2}$$

and

$$p_0(t + h) = p_0(t)(1 - \lambda_0 h) + p_0(1 - \lambda_1 h)(\mu_1 h).$$

Therefore

$$\frac{p_n(t + h) - p_n(t)}{h} = -p_n(t)[\lambda_n + \mu_n - \lambda\mu_n h] + p_n - (t)(\lambda_{n-1} - \lambda_{n+1}\mu_{n-1} h) + p_n + (t)(\mu_{n+1} - \lambda_{n+1}\mu_{n+1} h) \tag{A1-3}$$

and

$$\frac{p_0(t + h) - p_0(t)}{h} = -p_0(t)\lambda + p_1(\mu_1 - \lambda\mu_1 h). \tag{A1-4}$$

The differential-difference equations are now determined as:

$$\lim_{h\to 0} \frac{p_n(t+h) - p_n(t)}{h} = -p_n(t)(\lambda_n + \mu_n) + p_{n-1}(t)\lambda_{n-1} + p_{n+1}(t)\mu_{n+1}$$

and

$$\lim_{h\to 0} \frac{p_0(t+h) - p_0(t)}{h} = -p_0(t)\lambda_0 + p_1(t)\mu_1$$

such that

$$-p_n(\lambda_n + \mu_n) + \lambda_{n-1}p_{n-1} + \mu_{n+1}p_{n+1} = 0 \qquad \text{(A1-5)}$$

and

$$-\lambda_0 p_0 + \mu_1 p_1 = 0. \qquad \text{(A1-6)}$$

p_{n+1} and p_1 are determined to be

$$p_{n+1} = \frac{\lambda_n + \mu_n}{\mu_{n+1}} p_n - \frac{\lambda_{n-1}}{\mu_{n+1}} p_{n-1} \qquad \text{(A1-7)}$$

and

$$p_1 = \frac{\lambda_0}{\mu_1} p_0. \qquad \text{(A1-8)}$$

Substituting A1-8 into A1-7,

$$p_2 = \frac{\lambda_1 + \mu_1}{\mu_2} p_1 + \frac{\lambda_0}{\mu_2} p_0 = \frac{\lambda_0\lambda_1}{\mu_1\mu_2} p_0. \qquad \text{(A1-9)}$$

Substituting A1-9 into A1-7,

$$p_3 = \frac{\lambda_0\lambda_1\lambda_2}{\mu_1\mu_2\mu_3} p_0. \qquad \text{(A1-10)}$$

By induction

$$p_n = \frac{\lambda_0\lambda_1\ldots\mu_{n-1}}{\mu_1\mu_2\ldots\mu_n} p_0 = p_0 \prod_{i=1}^{n} \frac{\lambda_{i-1}}{\mu_i}. \qquad \text{(A1-11)}$$

Substituting A1-1 into A1-11,

$$p_n = \begin{cases} \frac{\lambda^n}{n!\mu^n} p_0 & (1 \leq n \leq c) \\ \frac{\lambda^n}{c^{n-c}c!\mu^n} p_0 & (n \geq c) \end{cases} \tag{A1-12}$$

Using the fact that $\sum_{h=0}^{\infty} p_n = 1$, p_0 can be determined by

$$p_0[\sum_{n=0}^{c-1} \frac{\lambda^n}{n!\mu^n} + \sum_{n=c}^{\infty} \frac{\lambda^n}{c^{n-c}c!\mu^n}] = 1. \tag{A1-13}$$

For simplicity of notation, let $r = \lambda/\mu$ and $\rho = r/c = \lambda/c\mu$. Equation A1-13 becomes

$$p_0[\sum_{n=0}^{c-1} \frac{r^n}{n!} + \sum_{n=c}^{\infty} \frac{r^n}{c^{n-c}c!}] = 1. \tag{A1-14}$$

The latter summation series in equation A1-14 is simplified as

$$\sum_{n=c}^{\infty} \frac{r^n}{c^{n-c}c!} = \frac{r^c}{c!} \sum_{n=c}^{\infty} (\frac{r}{c})^{n-c} = \frac{r^c}{c!} \sum_{m=0}^{\infty} (\frac{r}{c})^m$$

$$= \frac{r^c}{c!} \frac{1}{(1 - r/c)}, \ (r/c = \rho < 1), \tag{A1-15}$$

recognizing that $\sum_{m=0}^{\infty} (\frac{r}{c})^m$ is a geometric series with the sum $\frac{1}{(1 - r/c)}$. By substitution,

$$p_0 = [\sum_{n=0}^{c-1} \frac{r^c}{n!} + \frac{cr^c}{c!(c - r)}]^{-1}, \ (r/c = \rho < 1)$$

$$= [\sum_{n=0}^{c-1} \frac{1}{n!} (\frac{\lambda}{\mu})^n + \frac{1}{c!} (\frac{\lambda}{\mu})^c (\frac{c\mu}{c\mu - \lambda})]^{-1}. \tag{A1-16}$$

Measures of effectiveness are now derivable having determined the probability of zero messages in the system during steady state.

The first measure of effectiveness of concern is the expected queue size, Lg. By definition, the total length of queues is equivalent to the number in the system reduced by the number of servers. Therefore,

$$Lg = \sum_{n=c}^{\infty} (n - c)p_n$$

$$= \sum_{n=c}^{\infty} \frac{n}{c^{n-c}c!} r^n p_0 - \sum_{n=c}^{\infty} \frac{c}{c^{n-c}c!} r^n p_0. \tag{A1-17}$$

Again using the sum of a geometric series,

$$Lg = p_0 \frac{r^{c+1}/c}{c!} \left[\frac{1}{(1 - r/c)^2} + \frac{c^2/r}{1 - r/c} - \frac{c^2/r}{1 - r/c}\right]$$

$$= \left[\frac{r^{c+1}/c}{c!(1 - r/c)^2}\right] p_0 = \left[\frac{(\lambda/\mu)^c \lambda\mu}{(c - 1)!(c\mu - \lambda)^2}\right] p_0. \tag{A1-18}$$

Applying Little's formula (31) the expected waiting time is

$$Wg = \frac{Lg}{\lambda} = \left[\frac{(\lambda/\mu)^c \mu}{(c - 1)!(c\mu - \lambda)^2}\right] p_0, \tag{A1-19}$$

the waiting time in the system is

$$W = \frac{1}{\mu} + \left[\frac{(\lambda/\mu)^c \mu}{(c - 1)!(c\mu - \lambda)^2}\right] p_0,$$

and finally, the expected number in the system is

$$L = \lambda W = \frac{\lambda}{\mu} + \left[\frac{(\lambda/\mu)^c \lambda\mu}{(c - 1)!(c\mu - \lambda)^2}\right] p_0$$

$$= r + \left[\frac{r^{c+1}/c}{c!(1 - r/c)^2}\right] p_0. \tag{A1-20}$$

Though waiting time distributions are useful in some applications, their use for network analysis is limited. Reference is made to Gross and Harris (37) for their derivation.

APPENDIX B

The data contained in Appendix B supports the graphical presentations shown in Chapter V. Headings and titles make each table self explanatory and therefore little additional comment is necessary.

Table B1-1. Sixteen node network max hop/average hop data.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{j}$	$U_{.95}$	MAX HOP
1	0.2	6/2.60	6/2.55	6/2.57	6/2.58	2.55	2.57	2.60	6
	0.4	7/2.79	6/2.64	6/2.60	6/2.73	2.58	2.68	2.78	7
	0.6	7/2.62	6/2.55	7/2.70	6/2.60	2.55	2.62	2.62	7
	0.8	6/2.57	6/2.60	6/2.57	7/2.60	2.57	2.59	2.59	7
	1.0	6/2.52	6/2.58	6/2.51	6/2.59	2.51	2.56	2.56	6
2	0.2	6/2.60	6/2.55	6/2.57	6/2.58	2.55	2.57	2.60	6
	0.4	6/2.73	6/2.64	6/2.61	6/2.69	2.60	2.66	2.72	6
	0.6	6/2.61	6/2.54	6/2.69	6/2.60	2.54	2.62	2.69	6
	0.8	6/2.69	6/2.58	6/2.56	7/2.67	2.55	2.62	2.70	7
	1.0	6/2.57	12/2.60	7/2.60	6/2.61	2.58	2.60	2.62	12
3	0.2	6/2.60	6/2.55	10/4.16	6/2.58	2.12	3.05	3.98	10
	0.4	6/2.71	6/2.64	9/3.79	6/2.66	2.34	3.00	3.66	9
	0.6	6/2.61	6/2.54	6/2.60	6/2.68	2.55	2.62	2.69	6
	0.8	6/2.68	6/2.67	6/2.64	6/2.59	2.58	2.63	2.68	6
	1.0	6/2.66	12/2.85	25/4.00	6/2.61	2.31	3.08	3.85	25
4	0.2	6/2.60	12/3.63	6/2.69	6/2.58	2.23	2.83	3.42	12
	0.4	2/2.71	11/4.20	6/2.59	6/2.66	2.04	2.95	3.86	11
	0.6	6/2.61	9/3.77	6/2.59	6/2.68	2.19	2.86	3.54	9
	0.8	6/2.69	10/3.56	12/3.48	6/2.50	2.39	3.03	3.66	12
	1.0	6/2.58	7/3.16	31/3.83	6/2.61	2.39	3.08	3.77	31

Table B1-2. Twenty-five node network hop data.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{j}$	$U_{.95}$	MAX HOP
1	0.2	3.70	3.69	3.75	3.78	3.20	3.75	3.29	86
	0.4	3.89	3.84	3.76	3.77	3.72	3.79	3.87	75
	0.6	3.66	3.77	3.77	3.77	3.69	3.76	3.82	111
	0.8	3.67	3.67	3.65	3.66	3.65	3.66	3.67	61
	1.0	3.57	3.61	3.73	3.79	3.59	3.71	3.83	47
2	0.2	3.31	3.26	3.27	3.27	3.25	3.27	3.30	10
	0.4	3.38	3.36	3.36	3.36	3.35	3.36	3.37	9
	0.6	3.20	3.24	3.28	3.28	3.22	3.26	3.31	9
	0.8	3.27	3.29	3.32	3.31	3.28	3.31	3.33	10
	1.0	3.31	3.28	3.29	3.29	3.28	3.29	3.30	10
3	0.2	3.03	3.26	3.26	3.26	3.10	3.24	3.37	8
	0.4	3.23	3.29	3.30	3.29	3.25	3.29	3.32	9
	0.6	3.25	3.23	3.26	3.27	3.24	3.26	3.28	9
	0.8	3.35	3.35	3.36	3.35	3.35	3.35	3.36	9
	1.0	3.78	3.29	3.29	3.32	3.28	3.30	3.32	10
4	0.2	3.38	3.34	3.35	3.36	3.33	3.36	3.38	9
	0.4	3.44	3.40	3.40	3.39	3.37	3.40	3.43	11
	0.6	3.31	3.39	3.43	3.44	3.35	3.41	3.48	11
	0.8	3.50	3.59	3.44	3.47	3.41	3.49	3.56	10
	1.0	3.67	3.64	3.64	3.62	3.61	3.64	3.66	12

Table B1-3. Thirty-six node network max hop/average hop data.

		Average packets delivered per node							
SD	LF	50	100	150	200	$L_{.95}$	$\hat{j}$	$U_{.95}$	MAX HOP
1	0.2	15/4.43	15/4.42	178/4.45	150/4.43	4.42	4.43	4.45	178
	0.4	79/4.42	182/4.46	138/4.45	141/4.39	4.39	4.43	4.46	182
	0.6	15/4.34	150/4.44	150/4.43	150/4.51	4.37	4.46	4.54	150
	0.8	43/4.29	174/4.40	144/4.43	146/4.50	4.34	4.44	4.54	174
	1.0	64/4.26	86/4.45	91/4.36	113/4.70	4.28	4.50	4.73	113
2	0.2	10/3.88	10/3.90	10/3.92	10/3.93	3.89	3.92	3.94	10
	0.4	10/3.96	10/3.95	10/3.90	10/3.96	3.91	3.94	3.97	10
	0.6	10/3.90	10/3.93	10/3.92	10/3.94	3.91	3.93	3.95	10
	0.8	10/3.94	10/3.97	10/3.97	10/3.94	3.93	3.96	3.98	10
	1.0	10/3.89	10/3.97	34/3.94	29/3.95	3.90	3.95	3.99	34
3	0.2	12/3.87	12/3.91	12/3.87	12/3.90	3.87	3.89	3.91	12
	0.4	10/3.94	12/3.92	12/3.90	12/3.93	3.90	3.92	3.94	12
	0.6	12/3.89	12/3.91	12/3.91	12/3.92	3.90	3.91	3.93	12
	0.8	12/3.93	11/3.92	12/3.94	46/3.94	3.92	3.94	3.95	46
	1.0	26/3.96	50/4.05	14/4.00	36/4.04	3.97	4.02	4.07	50
4	0.2	12/3.87	11/3.91	12/3.87	12/3.88	3.86	3.88	3.90	12
	0.4	11/3.91	12/3.88	12/3.88	11/3.90	3.87	3.89	3.91	12
	0.6	11/3.88	13/3.91	13/3,90	13/3.92	3.89	3.91	3.93	13
	0.8	10/3.88	12/3.94	45/3.92	13/3.93	3.89	3.92	3.95	45
	1.0	10/3.95	36/4.00	49/4.04	50/4.05	3.97	4.03	4.08	50

Table B1-4. Sixteen node network queue length/observation data.

SD	LF	Average packets delivered per node: 50	100	150	200	$L_{.95}$	$\hat{q}$	$U_{.95}$
1	0.2	1.92/5575	1.89/11074	1.89/16729	1.90/22326	1.89	1.91	1.92
	0.4	2.32/5573	2.24/10675	2.25/15874	2.31/21900	2.23	2.28	2.33
	0.6	2.59/5195	2.64/10139	2.70/15934	2.69/20587	2.61	2.67	2.73
	0.8	3.03/5057	3.32 10223	3.35/15139	3.25/20425	3.10	3.27	3.44
	1.0	3.63/4952	3.84/10121	4.01/15022	3.98/20216	3.72	3.93	4.13
2	0.2	1.89/5575	1.87/11073	1.87/16731	1.88/22327	1.86	1.88	1.89
	0.4	2.23/5475	2.22/10676	2.23/15844	2.25/21675	2.22	2.24	2.25
	0.6	2.48/5193	2.53/10106	2.64/15874	2.61/20592	2.50	2.59	2.68
	0.8	2.96/5245	3.03/10178	3.26/15169	3.14/20817	2.98	3.14	3.29
	1.0	3.71/5046	3.85/10208	4.50/15298	4.00/20354	3.69	4.09	4.50
3	0.2	1.90/5575	1.87/11071	2.33/24208	1.88/22323	1.79	2.05	2.32
	0.4	2.22/5438	2.22/10671	2.91/21500	2.27/21510	2.15	2.52	2.90
	0.6	2.57/5207	2.52/10096	2.59/15454	2.62/21110	2.54	2.59	2.64
	0.8	2.95/5255	3.09/10421	3.01/15491	3.24/30320	2.98	3.13	3.28
	1.0	4.95/5176	4.76/11001	15.0/22557	3.98/20339	2.26	8.42	14.58
4	0.2	1.90/5575	2.19/14473	1.93/17081	1.88/22324	1.80	1.97	2.14
	0.4	2.22/5432	3.35/15693	2.23/15750	2.25/21413	1.91	2.57	3.23
	0.6	2.56/5208	3.94/14082	2.67/15457	2.64/21120	2.19	2.97	3.74
	0.8	2.99/5291	5.66/13388	5.18/19638	3.29/20346	2.86	4.44	6.01
	1.0	3.99/5069	4.75/12013	13.88/22649	4.10/20353	2.25	7.91	13.51

Table B1-5. Twenty-five node network queue length/observation data.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{q}$	$U_{.95}$
1	0.2	2.05/11711	2.03/23569	2.06/35890	2.07/48094	2.04	2.06	2.08
	0.4	2.48/11637	2.47/23079	2.47/33923	2.48/45170	2.47	2.47	2.48
	0.6	2.86/10935	2.97/22182	2.94/33272	2.94/44322	2.88	2.94	2.99
	0.8	3.40/10704	3.36/21422	3.33/32116	3.32/42940	3.30	3.34	3.38
	1.0	4.00/10478	4.20/21152	4.79/32902	4.84/43879	4.12	4.62	5.11
2	0.2	2.00/10810	1.98/21472	1.99/32251	1.99/43021	1.98	1.99	2.00
	0.4	2.41/10342	2.37/20508	2.36/30810	2.39/41051	2.35	2.38	2.41
	0.6	2.77/9717	2.82/19518	2.82/29544	2.81/39496	2.78	2.81	2.84
	0.8	3.50/9878	3.44/19522	3.42/29603	3.40/39326	3.37	3.42	3.47
	1.0	4.07/9840	3.79/19392	3.99/29259	4.01/39002	3.82	3.97	4.11
3	0.2	2.00/10789	1.97/21434	1.98/32213	1.98/42878	1.97	1.98	1.99
	0.4	2.32/10213	2.35/20197	2.34/30351	2.37/40424	2.33	2.35	2.38
	0.6	2.75/9795	2.80/19442	2.84/29386	2.82/39280	2.77	2.82	2.86
	0.8	3.48/9918	3.40/19877	3.52/29920	3.43/39747	3.39	3.46	3.52
	1.0	3.55/9780	4.03/19550	4.29/29281	4.56/39190	3.76	4.26	4.76
4	0.2	1.99/10980	1.98/21857	1.98/32883	1.99/43859	1.98	1.99	1.99
	0.4	2.38/10513	2.37/20825	2.36/31131	2.41/41457	2.36	2.38	2.41
	0.6	2.77/9997	2.86/20525	2.92/30651	2.89/41082	2.81	2.88	2.96
	0.8	3.54/10914	3.49/20474	3.57/30508	3.51/40857	3.49	3.53	3.57
	1.0	5.39/10914	5.16/21319	5.46/31941	5.33/42267	5.19	5.34	5.49

Table B1-6. Thirty-six node network queue length/observation data.

		Average packets delivered per node						
SD	LF	50	100	150	200	$L_{.95}$	$\hat{q}$	$U_{.95}$
1	0.2	2.12/20368	2.13/40223	2.15/60267	2.15/80402	2.13	2.14	2.16
	0.4	2.56/18824	2.55/38162	2.54/56783	2.55/75056	2.54	2.55	2.56
	0.6	2.93/18585	2.98/37170	3.00/55921	3.05/75362	2.95	3.01	3.07
	0.8	3.56/18310	3.64/36425	3.84/54977	4.01/74717	3.64	3.84	4.04
	1.0	4.23/18068	5.02/36617	4.97/54187	6.79/78036	4.37	5.64	6.92
2	0.2	2.04/18187	2.05/36474	2.08/54370	2.07/73001	2.05	2.09	2.09
	0.4	2.40/17155	2.42/34341	2.40/50903	2.41/68760	2.40	2.41	2.42
	0.6	2.73/16631	2.78/33445	2.79/50072	2.79/62055	2.75	2.78	2.87
	0.8	3.22/16687	3.51/33322	3.39/49900	3.50/66179	3.28	3.44	3.60
	1.0	3.67/16449	4.98/33191	4.66/49349	4.43/65861	3.88	4.53	5.19
3	0.2	2.04/18147	2.08/36304	2.05/54450	2.06/72440	2.04	2.06	2.08
	0.4	2.41/17121	2.41/33992	2.43/50897	2.42/68338	2.40	2.42	2.43
	0.6	2.71/16596	2.76/33341	2.77/49890	2.80/66820	2.73	2.77	2.82
	0.8	3.37/16695	3.59/32903	3.57/49588	3.57/66185	3.43	3.55	3.68
	1.0	4.32/16798	5.54/33818	4.75/50008	5.24/67340	4.43	5.06	5.70
4	0.2	2.04/18175	2.07/36465	2.05/54456	2.06/72490	2.04	2.06	2.07
	0.4	2.32/17083	2.42/33680	2.43/50756	2.41/67815	2.35	2.40	2.47
	0.6	2.74/16653	2.79/33320	2.79/49866	2.80/66788	2.76	2.79	2.82
	0.8	3.23/16500	3.30/33209	3.42/49451	3.54/65980	3.26	3.47	3.59
	1.0	4.19/16637	4.75/33358	5.05/50448	4.82/67405	4.38	4.81	5.24

Table B1-7. Sixteen node network average delay data.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{d}$	$U_{.95}$
1	0.2	.1763	.1718	.1735	.1736	.1713	.1735	.1757
	0.4	.2129	.1948	.1963	.2057	.1914	.2041	.2114
	0.6	.2107	.2104	.2261	.2164	.2089	.2175	.2262
	0.8	.2387	.2650	.2597	.2521	.2422	.2556	.2691
	1.0	.2736	.2938	.3096	.3211	.2829	.3035	.3241
2	0.2	.1745	.1711	.1724	.1732	.1710	.1727	.1743
	0.4	.2043	.1964	.1943	.2005	.1930	.1982	.2034
	0.6	.2088	.2045	.2237	.2149	.2051	.2149	.2246
	0.8	.2469	.2395	.2582	.2578	.2425	.2532	.2638
	1.0	.2946	.3098	.3800	.3277	.2927	.3365	.3803
3	0.2	.1750	.1716	.3036	.1732	.1355	.2122	.2889
	0.4	.2000	.1965	.3327	.2024	.1617	.2401	.3184
	0.6	.2175	.2049	.2153	.2211	.2076	.2158	.2239
	0.8	.2457	.2560	.2445	.2606	.2441	.2534	.2626
	1.0	.4591	.4446	2.254	.3265	-.1453	.9416	2.0286
4	0.2	.1755	.2571	.1834	.1736	.1463	.1934	.2405
	0.4	.2001	.4377	.1946	.2000	.1050	.2459	.3868
	0.6	.2119	.4628	.2229	.2241	.1271	.2703	.4134
	0.8	.2520	.6607	.5771	.2665	.1894	.4371	.6847
	1.0	.3311	.4738	1.772	.3411	-.0253	.7959	1.6172

Table B1-8. Twenty-five node network average delay data.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{d}$	$U_{.95}$
1	0.2	.2516	.2488	.2540	.2563	.2499	.2536	.2574
	0.4	.2916	.2869	.2823	.2834	.2797	.2846	.2482
	0.6	.2987	.3207	.3131	.3187	.3037	.3154	.2974
	0.8	.3460	.3437	.3396	.3401	.3377	.3413	.3549
	1.0	.4022	.4351	.5045	.5324	.4207	.4916	.4936
2	0.2	.2250	.2199	.2208	.2212	.2186	.2212	.2275
	0.4	.2590	.2516	.2508	.2528	.2482	.2526	.2605
	0.6	.2661	.2762	.2777	.2777	.2697	.2762	.3140
	0.8	.3371	.3359	.3355	.3343	.3339	.3353	.3707
	1.0	.3939	.3616	.3938	.4012	.3695	.3903	.6698
3	0.2	.2238	.2187	.2192	.2194	.2169	.2196	.2574
	0.4	.2472	.2464	.2458	.2481	.2458	.2470	.2895
	0.6	.2692	.2733	.2974	.2799	.2677	.2826	.3271
	0.8	.3499	.3387	.3547	.3439	.3385	.3467	.3448
	1.0	.3393	.3979	.4245	.4706	.3646	.4291	.5624
4	0.2	.2281	.2244	.2254	.2260	.2239	.2257	.2238
	0.4	,2601	.2554	.2567	.2594	.2552	.2579	.2570
	0.6	.2765	.2924	.3052	.3025	.2834	.2987	.2828
	0.8	.3690	.3589	.3682	.3652	.3597	.3652	.3366
	1.0	.5685	.6185	.6532	.6286	.5861	.6280	.4111

Table B1-9. Thirty-six node network average delay data.

		Average packets delivered per node						
SD	LF	50	100	150	200	$L_{.95}$	$\hat{d}$	$U_{.95}$
1	0.2	.3012	.2975	.3506	.3004	.2980	.3000	.3019
	0.4	.3325	.3344	.3327	.3296	.3294	.3318	.3341
	0.6	.3541	.3720	.3718	.3851	.3604	.3754	.3904
	0.8	.4193	.4325	.4702	.4956	.4266	.4677	.5088
	1.0	.4843	.6355	.6077	.9455	.5052	.7360	.9668
2	0.2	.2572	.2606	.2636	.2645	.2588	.2627	.2666
	0.4	.2909	.2912	.2871	.2900	.2873	.2895	.2917
	0.6	.3133	.3201	.3216	.3221	.3159	.3207	.3255
	0.8	.3582	.4024	.3845	.3935	.3666	.3891	.4115
	1.0	.4015	.5969	.5470	.5159	.5075	.5300	.5524
3	0.2	.2584	.2643	.2591	.2619	.2580	.2612	.2644
	0.4	.2931	.2900	.2913	.2914	.2898	.2913	.2928
	0.6	.3136	.3909	.3203	.3252	.2835	.3357	.3780
	0.8	.3869	.4142	.4119	.4172	.3956	.4120	.4284
	1.0	.5179	.6991	.5146	.6661	.5326	.6304	.7283
4	0.2	.2607	.2636	.2604	.2617	.2599	.2616	.2633
	0.4	.2842	.2900	.2913	.2906	.2862	.2901	.2739
	0.6	.3197	.3241	.3245	.3259	.3214	.3246	.3279
	0.8	.3675	.3798	.3923	.4089	.3731	.3940	.4148
	1.0	.4910	.5812	.6342	.6034	.5245	.5970	.6694

Table B1-10. Sixteen node network utilization data.

		Average packets delivered per node						
SD	LF	50	100	150	200	$L_{.95}$	$\hat{\rho}$	$U_{.95}$
1	0.2	.1920	.1250	.1257	.1263	.0934	.1324	.1715
	0.4	.3294	.2805	.2998	.3145	.2802	.3048	.3294
	0.6	.4696	.4825	.5349	.5176	.4753	.5110	.5466
	0.8	.6855	.8294	.8386	.7983	.7225	.8053	.8882
	1.0	.8889	1.155	1.201	1.206	.9850	1.1626	1.3402
2	0.2	.1277	.1245	.1249	.1260	.1239	.1255	.1272
	0.4	.3174	.2828	.2967	.3066	.2826	.3000	.3173
	0.6	.4653	.4691	.5292	.5140	.4670	.5047	.5424
	0.8	.7099	.7431	.9318	.8156	.7098	.8254	.9410
	1.0	1.122	1.186	1.534	1.262	1.0991	1.3144	1.5297
3	0.2	.1280	.1249	.2389	.1260	.0936	.1599	.2261
	0.4	.3108	.2830	.5292	.3123	.2368	.3714	.5059
	0.6	.4741	.4698	.5120	.5252	.4727	.5051	.5374
	0.8	.7237	.8003	.7686	.8244	.7416	.7928	.8440
	1.0	1.761	1.689	8.723	1.284	-.5659	3.6444	7.8547
4	0.2	.1284	.1936	.1459	.1263	.1090	.1459	.1827
	0.4	.3109	.6941	.3062	.3086	.1584	.3852	.6120
	0.6	.4728	1.107	.5069	.5323	.2778	.6337	.9895
	0.8	.7420	2.151	1.861	.8434	.5628	1.4001	2.2373
	1.0	1.281	1.871	6.957	1.343	-.0995	3.1266	6.3527

Table B1-11. Twenty-five node network utilization data.

SD	LF	Average packets delivered per node						
		50	100	150	200	$L_{.95}$	$\hat{\rho}$	$U_{.95}$
1	0.2	.1530	.1484	.1524	.1527	.1492	.1518	.1543
	0.4	.3512	.3470	.3465	.3527	.3430	.3486	.3541
	0.6	.5673	.6161	.5965	.6006	.5752	.5991	.6231
	0.8	.8874	.8730	.8565	.8539	.8435	.8619	.8802
	1.0	1.2244	1.3475	1.5746	1.6859	1.2916	1.5387	1.7858
2	0.2	.1368	.1312	.1325	.1331	.1301	.1329	.1357
	0.4	.3228	.3164	.3155	.3213	.3145	.3187	.3230
	0.6	.5032	.5277	.5244	.5230	.5093	.5224	.5354
	0.8	.8630	.8580	.8605	.8500	.8494	.8561	.8627
	1.0	1.973	1.0781	1.1641	1.1762	1.0934	1.1551	1.2167
3	0.2	.1360	.1305	.1315	.1320	.1291	.1320	.1348
	0.4	.3081	.3100	.3092	.3153	.3079	.3117	.3155
	0.6	.5135	.5247	.5386	.5326	.5181	.5309	.5437
	0.8	.9184	.8779	.9295	.8905	.8743	.9025	.9306
	1.0	1.0294	1.2346	1.3417	1.4954	1.1202	1.3505	1.5809
4	0.2	.1387	.1340	.1352	.1360	.1333	.1356	.1380
	0.4	.3242	.3213	.3229	.3297	.3211	.3254	.3297
	0.6	.5239	.5587	.5764	.5695	.5374	.5649	.5923
	0.8	.9337	.9114	.9364	.9197	.9106	.9245	.9383
	1.0	1.7727	1.9244	2.0464	1.9730	1.8291	1.9653	2.1014

Table B1-12. Thirty-six node network utilization data.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{\rho}$	$U_{.95}$
1	0.2	.1487	.1469	.1645	.1526	.1453	.1546	.1640
	0.4	.3423	.3404	.3466	.3522	.3410	.3472	.3533
	0.6	.5223	.5640	.5628	.5886	.5370	.5693	.6016
	0.8	.8455	.9059	.9909	1.060	.8761	.9870	1.0979
	1.0	1.223	1.687	1.609	2.506	1.3102	1.9448	2.5794
2	0.2	.1279	.1287	.1437	.1371	.1277	.1365	.1453
	0.4	.2993	.3034	.3035	.3016	.3000	.3023	.3046
	0.6	.4622	.4851	.4867	.4923	.4705	.4862	.5018
	0.8	.7154	.8350	.8073	.8415	.7489	.8173	.8858
	1.0	.9884	1.596	1.438	1.334	1.0802	1.3830	1.6859
3	0.2	.1275	.1367	.1290	.1376	.1260	.1337	.1405
	0.4	.3181	.3096	.3096	.3060	.3030	.3090	.3151
	0.6	.4630	.5926	.4849	.4971	.4418	.5091	.5765
	0.8	.7820	.8676	.8910	.8802	.8126	.8711	.9296
	1.0	1.286	1.859	1.525	1.749	1.3588	1.6575	1.9562
4	0.2	.1287	.1346	.1297	.1330	.1286	.1319	.1352
	0.4	.2426	.13060	.3095	.3036	.2621	.2998	.3374
	0.6	.4721	.4925	.4913	.4981	.4790	.4923	.5057
	0.8	.7257	.7633	.8359	.8816	.7460	.8286	.9113
	1.0	1.201	1.554	1.669	1.587	1.3230	1.5664	1.8098

Table B1-13. Supporting data for three algorithms evaluated.

Algorithm	LF	Average hops	Maximum hops	$\bar{q}$	$\bar{d}$	$\bar{\rho}$
CAT_1	0.6	3.94	10	2.79	.3221	.4923
	0.8	3.94	10	3.50	.3935	.8415
	1.0	3.95	29	4.43	.5159	1.3338
CAT_2	0.6	4.51	150	3.05	.3851	.5886
	0.8	4.50	146	4.01	.4956	1.0599
	1.0	4.69	113	6.79	.9455	2.5060
CAT_3	0.6	3.93	10	2.76	.3155	.4868
	0.8	3.97	10	3.49	.3943	.8407
	1.0	3.94	10	4.20	.4748	1.2316

Table B1-14. Confidence interval data.

Average packets delivered per node

Utilization factor

LF	50	100	150	200	$L_{.95}$	$\hat{\rho}$	$U_{.95}$
0.2	.1286	.1293	.1297	.1329	.1285	.1308	.1330
0.4	.2974	.3011	.3065	.3099	.2933	.3059	.3124
0.6	.4604	.5371	.5031	.4968	.4661	.5031	.5401
0.8	.7254	.7992	.8660	.8407	.7561	.8285	.9088
1.0	.9650	1.1736	1.2817	1.2316	1.0446	1.2084	1.3721

Delay

						$\hat{d}$	
0.2	.2606	.2618	.2604	.2615	.2603	.2611	.2619
0.4	.2890	.2891	.2884	.2915	.2882	.2898	.2914
0.6	.3122	.3318	.3163	.3155	.3084	.3187	.3290
0.8	.3579	.3754	.4047	.3943	.3656	.3900	.4144
1.0	.3939	.4449	.4950	.4748	.4151	.4668	.5185

Queue length

						$\hat{j}$	
0.2	2.042	2.043	2.043	2.063	2.040	2.047	2.053
0.4	2.378	2.393	2.430	2.403	2.381	2.407	2.432
0.6	2.740	2.871	2.781	2.763	2.762	2.788	2.814
0.8	3.260	3.388	3.502	3.491	3.319	2.451	3.583
1.0	3.620	4.060	4.341	4.203	3.790	4.158	4.525

Table B1-15. Supporting data for comparison curves.

SD	LF	Average packets delivered per node 50	100	150	200	$L_{.95}$	$\hat{d}$	$U_{.95}$
Algorithm 3 using 9.6 kbps circuits								
2	0.2	.2606	.2618	.2604	.2615	.2603	.2611	.2619
	0.4	.2890	.2891	.2884	.2915	.2882	.2898	.2915
	0.6	.3122	.3318	.3163	.3155	.3084	.3187	.3290
	0.8	.3579	.3754	.4047	.3943	.3656	.3900	.4144
	1.0	.3939	.4449	.4950	.4748	.4151	.4668	.5185
Algorithm 3 using 50.0 kbps circuits								
	0.2	.0588	.0582	.0584	.0584	.0581	.0584	.0587
	0.4	.0629	.0617	.0620	.0620	.0613	.0620	.0628
	0.6	.0675	.0664	.0700	.0686	.0667	.0685	.0703
	0.8	.0730	.0760	.0786	.0771	.0741	.0769	.0797
	1.0	.0854	.0879	.0906	.1028	.0853	.0944	.1035

Average queue - 2.9657

Table B1-16. 256 node network data.

	Average hops (avg msgs/node)						
(load factor)	25	50	75	100	$L_{.95}$	$\hat{h}$	$U_{.95}$
0.2	10.45	10.47	10.48	10.47	10.46	10.47	10.49
0.4	10.58	10.54	10.55	10.54	10.52	10.55	10.57
0.6	10.56	10.53	10.53	10.53	10.52	10.53	10.55
0.8	10.43	10.52	10.54	10.51	10.46	10.51	10.57
1.0	10.54	10.51	10.54	10.51	10.50	10.52	10.54
1.5	10.36	10.44	10.48	10.47	10.39	10.46	10.52
	Average queue					$\hat{q}$	
0.2	2.350	2.355	2.358	2.360	2.352	2.357	2.362
0.4	2.637	2.650	2.646	2.773	2.620	2.696	2.722
0.6	2.938	2.960	2.961	2.963	2.945	2.959	2.973
0.8	3.223	3.288	3.295	3.291	3.244	3.285	3.325
1.0	3.567	3.614	3.621	3.622	3.585	3.616	3.647
1.5	4.341	4.468	4.496	4.489	4.386	4.472	4.557
	Average delay					$\hat{d}$	
0.2	2.717	2.713	2.717	2.714	2.713	2.715	2.717
0.4	2.784	2.716	2.777	2.773	2.770	2.776	2.281
0.6	2.820	2.812	2.812	2.809	2.860	2.812	2.817
0.8	2.820	2.852	2.860	2.849	2.829	2.850	2.870
1.0	2.886	2.891	2.901	2.890	2.885	2.893	2.900
1.5	2.832	2.744	2.718	2.717	2.671	2.734	2.798
	Average utilization					$\hat{\rho}$	
0.2	.1278	.1311	.1328	.1331	.1292	.1321	.1349
0.4	.2692	.2724	.2728	.2732	.2704	.2725	.2747
0.6	.4072	.4175	.4170	.4168	.4102	.4160	.4219
0.8	.5345	.5577	.5616	.5614	.5427	.5580	.5733
1.0	.6844	.6874	.7107	.7096	.6863	.7029	.7194
1.5	1.0425	1.0878	1.1026	1.1005	1.0597	1.0927	1.1256

APPENDIX C

Appendix C contains a listing of the network simulator program used in support of research for this dissertation. The user's guide preceeding the expanded variable definition list and listing explains the various options available and the type and format of the input parameters. The program is sufficiently well documented that those versed in FORTRAN should experience no difficulty interpreting the event sequence within the program.

1. User's Guide

a. Description of Input Parameters

(1) Simulator Input Requirements

(a) NUMNOD - Establishes the number of nodes to be simulated. NUMNOD must meet the following restrictions: NUMNOD = N^2, $2 \leq N \leq 32$.

(b) MINLK - Determines the minimum number of links possessed by each node.

(c) MAXLK - Determines the maximum number of links allowed each node.

(d) NBUF - Establishes the maximum number of buffers at each node.

(e) MSGLVL - Establishes the maximum number of messages allowed active in the network at one time. The appropriate value of MSGLVL is derived as a function of n where NSET(n) is the GASP event and entity storage array.

(f) MAXSPL - Establishes the number of messages to be delivered for a given simulation run.

(g) LKAHD - Determines the search depth value to be simulated. LKAHD is constrained by $1 \leq$ LKAHD ≤ 4; all values greater than 4 are converted to 4.

(h) NDIGIT - A load factor synchronization parameter. Adjust NDIGIT (used in subrouting OTPUT) during simulator tuning to give an approximate 0.2 measured load factor for an input RE = 0.2. Subsequent runs

with increasingly higher RE's should result in measured load factors approximating RE.

(i) NOPRT - A print/no print flag for printing nodal coordinates, adjacency matrix, density matrix, etc.

NOPRT $\leq$ 0: do not print

NOPRT > 0: print

(j) ITEM - A threshold for generating interim output. For a given ITEM value, say I, statistics will be printed for every I message generated and for every I messages delivered.

(k) IROUT - A flag to prevent generation of minimum delay vectors.

IROUT = 0: do not generate MDV's

IROUT $\neq$ 0: generate MDV's

(l) NRFM - A hop inhibit threshold. The hop suppress mechanism will be implemented for LKAHD hops whenever hops = 2*NRFM*SQRT(NUMNOD).

(m) RE - A load factor starting parameter.

(n) REINC - An increment value for RE when executing two or more runs.

(o) UPTINT - Establishes the update interval for generating update vectors.

(p) XLN - Determines the user packet length.

(q) BIAS - A means of weighting the effects of the heuristic function on the evaluation of total delay.

(r) DNSIPT - A network loading variable for varying the interval of message generation. The following

values have been realized at RE = 0.2 using the simulator random number generator.

DNSIPT	Avg msg interval (in secs)
0.004	20.69
0.010	10.05
0.015	5.85
0.020	2.79
0.048	2.02
0.065	1.64
0.095	1.36

(2) GASP Input Requirements - Different GASP input is required for each network size to be simulated. A detailed description of GASP input parameters is provided by Pritzker (68) on pages 67-80.

b. Format for Simulator Input: Three types of input are possible depending upon the simulation options desired. The first two cards are always required.

(1) Format (12(2X,I8)): for inputting all integer values described in paragraph a(1) above.

(2) Format (6(2X,F8.3)): for inputting all real values described in paragraph a(1) above.

(3) Format (2(F5.2,2X),I5): Used for additional runs involving re-initialization of values for RE, REINC, and MAXSPL. This input is placed at the end of the GASP input parameters. It should be noted that the appropriate parameter in GASP (NNRNS) must reflect the total number of

runs desired including those with re-initialized simulator parameters.

c. Simulation Options: Table C1-1.

2. COMMON Variables in the Simulator

The following variable definitions are provided for ease of understanding program flow. Variable definitions are normally contained in comments to the program listing; however, in this particular case the number of variables to be defined warrant a separate composite reference. Each of the variables is expressed in the indicated user COMMON statements.

The definitions identify the routine of primary utilization as well as the purpose of each variable. No variable is used for a dual purpose.

a. COMMON/UCOM1/

(1) ADJMAT - adjacency matrix; used throughout the program, however, its elements are generated in the driver. Column 1 contains the numeric node designation, columns 2-5 indicate adjacent node connectivity, column 6 contains the number of adjacent nodes for the node identified in column 1, and column 7 is a running total of links.

(2) TRFMAT - traffic matrix; generated in the driver and used throughout the simulator. TRFMAT reflects the traffic density between each node and their adjacent nodes. It is used in INTLC to generate the mean interarrival rate for each node.

(3) CAPMAT - capacity matrix; generated in the driver to reflect the capacity of each link in the simulated network. Although links are set to 9.6 kbps in the driver, different capacities are possible with minor adjustments.

Table C1-1. Simulation options.

Requirement	RE	REINC	NNRNS	Comments
1 run, printout of node queues	as desired	0.0	1	Queue printout will only occur with NNRNS = 1
1 run, no printout of queues	as desired	0.0	2	Program will exit on error attempting to read a control card as data. Do not include data for format b(3).
n runs, incrementing RE with x for each run	as desired	x	n	Initial RE is the starting load factor; increment occurs only after the first run.
m sequences of n runs each incrementing RE with x for each run	as desired	x	m*n	RE, REINC, and MAXSPL may be respecified after each sequence. Include format b(3) data for m - 1 sequences.
1 run, load factor equal 1.0	1.0	0.0	2	Include data for format b(3) as though re-initializing but use the same parameter values used in data for format b(1).

(4) XMU - mean interarrival rate array; generated in INTLC by averaging the traffic densities of connected links obtained from TRFMAT.

(5) ITRACE - unused; originally intended as a message trace switch, however, storage requirements become excessive.

(6) DP - unused.

(7) XCORD - x coordinate array; generated in the driver and used in RTALGO to evaluate the weighted distance to the destination node.

(8) YCORD - y coordinate array; same as XCORD.

(9) Z - random number variable; used throughout the program.

(10) BSYLIN - busy line array; reflects the status (busy/not busy) of the respective interconnecting line. BSYLIN is used in several routines where availability of resources must be evaluated before releasing messages for transmission.

(11) MATHAG - used in RTALGO and PATHAG. MATHAG is an array of delay data to be compared for selecting the next (adjacent) node. It contains only the results of the cost evaluations to adjacent nodes in the current tree.

(12) LSORC - an array of cross references between source nodes and link numbers. LSORC is used in ENCODE and LINDEL to identify the most recent source of a message.

(13) LDEST - an array of cross references of link numbers to destinations. LDEST values provide indices to GASP for filing messages queued for transmission. It is used in ENCODE and LINDEL for obtaining destination node numbers.

(14) DISMAT - distance matrix; contains the distance from nodes to their adjacently connected nodes. DISMAT is established in the driver and used in GENMDV to reflect propagation delay in the minimum delay vector being generated.

(15) JQUE - input queue array; used in several routines to tabulate a count of messages awaiting for process by a node.

(16) IQUE - output queue array; used in several routines to tabulate a count of messages awaiting transmission over an outgoing link.

(17) NUMLIN - number of lines; a summation of ADJMAT(I,6) reflecting the total number of lines in the network. NUMLIN was originally used as a bound for testing that an erroneous line number was being used. Its use was later discontinued to conserve execution time.

(18) KSTOR - a temporary storage of the first index into MATHAG identifying the next node of least cost. KSTOR is used in RTALGO to pass the index back to CPUSVC.

(19) DELMAT - delay matrix; used in LINDEL to preserve the delay data contained in the minimum delay vector and in PATHAG to evaluate for minimum cost paths.

(20) MSGEVT - message event; used by GASP as an event filing parameter. MSGEVT is subsequently used by EVNTS to identify the type of event occurring. Its value is initialized in the driver.

(21) JSTIMP - just an imp event; same as MSGEVT except for imp inputs.

(22) IMPEVT - imp event; same as MSGEVT except for imp transmissions.

(23) NFNMTO - negative request for next message; same as MSGEVT.

(24) LEVENT - L event; same as MSGEVT for identifying a minimum delay vector generation event.

(25) NCKTO - acknowledge event; same as MSGEVT.

(26) KMITVT - transmission event; same as MSGEVT.

(27) INCDVT - incident event; same as MSGEVT to identify and file erred messages for retransmission.

(28) NODQUE - internal node queue files; same as MSGEVT.

(29) IMAT - integer matrix; used in earlier simulator testing. Its use was discontinued.

(30) IMATPT - same as IMAT.

(31) IVALS - same as IMAT.

(32) GMAT - same as IMAT.

(33) ITB - buffer array; used by several routines to tabulate buffer utilization. ITB contents are printed in OTPUT.

(34) LINMAT - line matrix; a matrix of line numbers between consecutively numbered nodes and adjacent nodes. LINMAT is generated in the driver and used in several routines as a cross-reference for re-identifying the immediate source node.

(35) XYLN - average link transmission interval; initialized in the driver to minimize execution time and used throughout the simulator.

(36) XZLN - average line transmission interval; same as XYLN.

(37) LKPT - same as IMAT.

(38) LINBSY - line busy; a variable for accumulating the number of times a needed line was found busy. LINBSY is altered in several routines and printed in subroutine OTPUT.

(39) NOBUFR - no buffer; a variable for accumulating the number of times a message was refused service because of insufficient node buffer space. Same as LINBSY.

(40) ICPBSY - CPU busy; used to accumulate the number of times service was refused because a node CPU was busy. Same as LINBSY.

b. COMMON/UCOM2/

(1) RELIM - use discontinued; originally used in MSG as a stopping parameter identified as the load factor limit.

(2) FIRST - a flag initialized to zero in the driver and incremented in INTLC to reflect the number of simulation runs made.

(3) RESAVE - a temporary storage variable for preserving the load factor. Its use was discontinued.

(4) UDTLIM - use discontinued.

c. COMMON/UCOM4/IOUT - messages delivered; IOUT is compared to MAXSPL in MSG and used as a stopping parameter which accumulates the number of messages delivered.

3. Listing of Simulator

```
C*************************************************************************00001000
C                                                                         00002000
C     MAIN                                                                00003000
C                                                                         00004000
C     AFTER GENERATING THE VARIOUS ARRAYS AND READING INPUT PARAMETERS,   00005000
C     MAIN CALLS GASP FOR A SIMULATION RUN.  ARRAYS AND OTHER VARIABLES   00006000
C     WHICH DO NOT HAVE TO BE RE-INITIALIZED AFTER EACH RUN SHOULD BE     00007000
C     INITIALIZED IN MAIN.  OTHERS SHOULD BE INITIALIZED IN SUBROUTINE    00008000
C     INTLC.                                                              00009000
C                                                                         00010000
C*************************************************************************00011000
C                                                                         00012000
      INTEGER ADJMAT,XCORD,YCORD                                          00013000
      REAL ITB,NBUF                                                       00014000
      DIMENSION NSET(86400)                                               00015000
      COMMON QSET(86400)                                                  00016000
C                                                                         00017000
C     GASP VARIABLES                                                      00018000
C                                                                         00019000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,       00020000
     1NCRDR,NNAPQ,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),    00021000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                              00022000
C                                                                         00023000
C                                                                         00024000
C     USER VARIABLES                                                      00025000
C                                                                         00026000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1   00027000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,     00028000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,  00029000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,5   00030000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,5   00031000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,      00032000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NREM,  00033000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY          00034000
      COMMON /UCOM2/ REINC,RELIM,FIRST,RESAVE,UDTLIM,NDIGIT               00035000
      COMMON /UCOM4/ IOUT                                                 00036000
C     THE MAIN EVENT STORAGE ARRAY IS REAL OR INTEGER.                    00037000
      EQUIVALENCE (NSET(1),QSET(1))                                       00038000
C                                                                         00039000
C     CONSISTENT WITH GASP, RENAME THE READER AND PRINTER.                00040000
C     TEST FOR SIMULATION LEVEL.                                          00041000
      NCRDR=5                                                             00042000
      NPRNT=6                                                             00043000
      READ(NCRDR,30,END=27)NUMNOD,MINLK,MAXLK,NBUF,MSGLVL,MAXSPL,         00044000
     1LKAHD,NDIGIT,NOPRT,ITEM,IROUT,NREM                                  00045000
      READ(NCRDR,32,END=27)RE,REINC,UPTINT,XLN,BIAS,DNSIPT                00046000
      WRITE(NPRNT,31)NUMNOD,MINLK,MAXLK,MBUF,LKAHD,BIAS,UDTINT,XLN,       00047000
     1MSGLVL,MAXSPL,RE,REINC                                              00048000
```

```
      Z = 1.0                                                          00049000
      TEMP = NUMNOD                                                    00050000
      N = SQRT(TEMP)                                                   00051000
      NREM = NREV*N*2                                                  00052000
C     INITIALIZE ADJACENCY, TRAFFIC, CAPACITY, DELAY, AND DISTANCE     00053000
C     MATRICES                                                         00054000
      DO 3 I=1,NUMNOD                                                  00055000
          DO 2 J=1,7                                                   00056000
              ADJMAT(I,J) = 0                                          00057000
2         CONTINUE                                                     00058000
3     CONTINUE                                                         00059000
      DO 5 I=1,NUMNOD                                                  00060000
          DO 4 J=1,5                                                   00061000
              TRFMAT(I,J) = 0.0                                        00062000
              CAPMAT(I,J) = 0.0                                        00063000
              DELMAT(I,J) = 0.0                                        00064000
              DISMAT(I,J) = 0.0                                        00065000
4         CONTINUE                                                     00066000
5     CONTINUE                                                         00067000
C     INITIALIZE MESSAGE INDICATOR POINTERS                            00068000
C     NAK AND NCNT ARE USED FOR OUTPUT COUNTERS.                       00069000
      NAK = 0                                                          00070000
      NCNT = 0                                                         00071000
C     HALFDN REQUIRED IN METHOD FOR DERIVING INDIVIDUAL NODE MESSAGE   00072000
C     GENERATION INTERVALS.                                            00073000
      HALFDN = DNSIPT/5.0                                              00074000
C     ESTABLISH VARIABLES FOR INDEXING APPROPRIATE GASP FILE IN NSET.  00075000
C     FIXING VALUES HERE SAVES EXECUTION TIME WHILE SIMULATING.        00076000
      RESAVE = RE                                                      00077000
      MSGEVT = NUMNOD+1                                                00078000
      JSTIMP = 2*NUMNOD+1                                              00079000
      IMPEVT = 3*NUMNOD+1                                              00080000
      NODQUE = 4*NUMNOD+1                                              00081000
      LEVENT = 5*NUMNOD+1                                              00082000
      NFNMTO = 6*NUMNOD+1                                              00083000
      NCKTO = 7*NUMNOD+1                                               00084000
      KMITVT = 8*NUMNOD                                                00085000
      INCDVT = 13*NUMNOD                                               00086000
      IGENCT = 0                                                       00087000
C     XYLN AND XZLN ARE ALSO TO DECREASE EXECUTION TIME.               00088000
      XYLN = XLN/256000.0                                              00089000
      XZLN = XLN/9600.0                                                00090000
C     ESTABLISH COORDINATES FOR PIVOT NODE AND REFLECT EXISTENCE       00091000
C     IN THE ADJACENCY MATRIX.                                         00092000
      XCORD(1) = 0                                                     00093000
      YCORD(1) = 0                                                     00094000
      ADJMAT(1,1) = 2                                                  00095000
      ADJMAT(1,6) = 1                                                  00096000
```

```
      DO 6 I=2,N                                                        00097000
          CALL RANDOM(Z)                                                00098000
C         ESTABLISH COORDINATES FOR THE FIRST ROW OF NODES ALONG        00099000
C         THE X AXIS WITH Y=0.                                          00100000
          XCORD(I) = XCORD(I-1)+(10.0*Z+200.0)                          00101000
          YCORD(I) = 0                                                  00102000
          ADJMAT(I,1) = I-1                                             00103000
          ADJMAT(I,6) = 1                                               00104000
C         NO POINTER TO THE RIGHT IF THIS IS THE LAST NODE IN THE ROW.  00105000
          IF(I.EQ.N)GO TO 10                                            00106000
          ADJMAT(I,2) = I+1                                             00107000
          ADJMAT(I,6) = ADJMAT(I,6)+1                                   00108000
6     CONTINUE                                                          00109000
      K = N                                                             00110000
C     GENERATE THE REMAINING COORDINATES, REFLECTING THEIR              00111000
C     CONNECTIVITY IN THE ADJACENCY MATRIX.                             00112000
      DO 13 J=2,N                                                       00113000
          DO 12 I=1,N                                                   00114000
              K = K+1                                                   00115000
7             CALL RANDOM(Z)                                            00116000
              XCORD(K) = XCORD(K-N)+(10.0*Z+200.0)                      00117000
              IF(I.EQ.1)GO TO 8                                         00118000
C             TEST FOR ILL-DEFINED X-COORDINATES.                       00119000
              IF(XCORD(K).LE.XCORD(K-1))GO TO 7                         00120000
              IF(I.EQ.N)GO TO 8                                         00121000
C             ANOTHER TEST FOR A POTENTIALLY BAD COORDINATE.            00122000
              IF(XCORD(K).GT.XCORD(K-N+1))GO TO 7                       00123000
8             CALL RANDOM(Z)                                            00124000
              YCORD(K) = YCORD(K-N)+(10.0*Z+200.0)                      00125000
              IF(J.LT.3)GO TO 9                                         00126000
C             TEST FOR ILL-DEFINED Y-COORDINATES.                       00127000
              IF(YCORD(K).LT.YCORD(K-N-1))GO TO 8                       00128000
9             MLIM = K-1                                                00129000
              DO 10 M = 1,MLIM                                          00130000
C                 TEST THE COMBINATION.  IN ALL TEST CASES, IF THE TEST 00131000
C                 FAILS THEN GENERATE ANOTHER COORDINATE.               00132000
                  IF((XCORD(M).EQ.XCORD(K)).AND.                        00133000
     1                (YCORD(M).EQ.YCORD(K)))GO TO 8                    00134000
10            CONTINUE                                                  00135000
              JPT = ADJMAT(K,6)+1                                       00136000
              ADJMAT(K,JPT) = K-N                                       00137000
              ADJMAT(K,6) = JPT                                         00138000
              KPT = ADJMAT(K-N,6)+1                                     00139000
              ADJMAT(K-N,KPT) = K                                       00140000
              ADJMAT(K-N,6) = KPT                                       00141000
              IF(I.EQ.1)GO TO 11                                        00142000
              JPT = JPT+1                                               00143000
              ADJMAT(K,JPT) = K-1                                       00144000
```

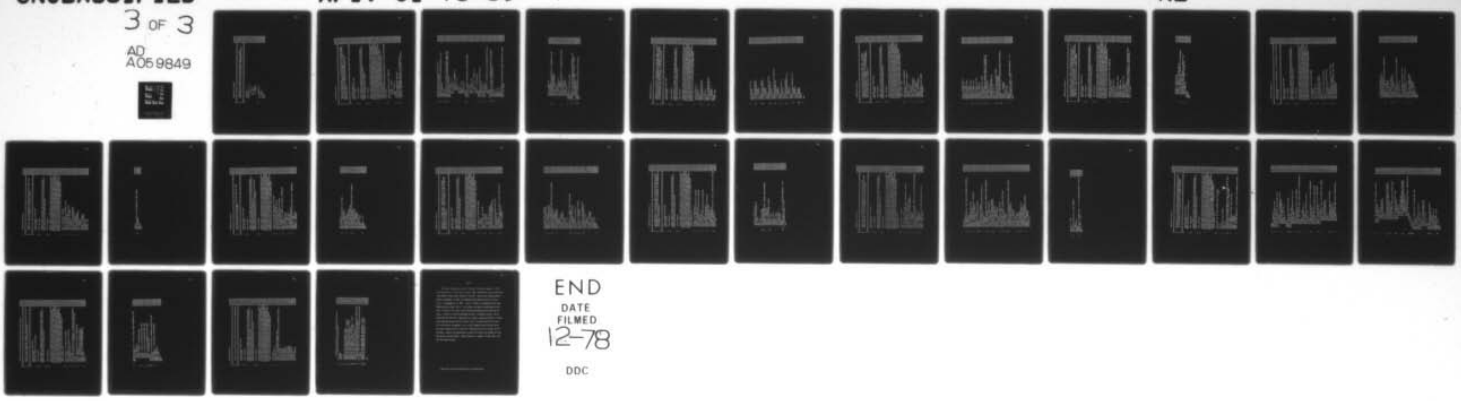
```
19            LKPT(ICTR,2) = ADJMAT(II,7)+JP                                    00193000
              LINMAT(I,J) = ICTR                                                00194000
              LSORC(ICTR) = I                                                   00195000
              LDEST(ICTR) = ADJMAT(I,J)                                         00196000
              ICTR = ICTR +1                                                    00197000
              IF(ICTR.GT.ADJMAT(I+1,7))GO TO 21                                 00198000
20        CONTINUE                                                              00199000
21    CONTINUE                                                                  00200000
C     TEST PRINT FLAG.  IF IT IS ON, PRINT MATRICES                             00201000
      IF(NOPRT)24,24,22                                                         00202000
22    WRITE(NPRNT,28)                                                           00203000
      DO 23 I=1,NUMNOD                                                          00204000
          WRITE(NPRNT,29)I,(ADJMAT(I,J),J=1,5),(CAPMAT(I,K),K=1,5),             00205000
     1(DISMAT(I,M),M=1,5),ADJMAT(I+1,7),XCORD(I),YCORD(I)                       00206000
23    CONTINUE                                                                  00207000
C     CONVERT THE DISTANCE MATRIX TO DELAY.                                     00208000
24    DO 26 I=1,NUMNOD                                                          00209000
          JMAX = ADJMAT(I,6)                                                    00210000
          DO 25 J=1,JMAX                                                        00211000
              DISMAT(I,J) = DISMAT(I,J)*0.0008+0.00007                          00212000
25        CONTINUE                                                              00213000
26    CONTINUE                                                                  00214000
C     FIRST IS A VARIABLE FOR RE-INITIALIZING FOR SUBSEQUENT                    00215000
C     RUNS.                                                                     00216000
      FIRST = 0.0                                                               00217000
C                                                                               00218000
C     BEGIN SIMULATION                                                          00219000
C                                                                               00220000
      CALL GASP                                                                 00221000
C                                                                               00222000
      GO TO 1                                                                   00223000
27    STOP                                                                      00224000
28    FORMAT('1',1X,'NODE',7X,'ADJACENT NODES',10X,'PORT CAPACITY',             00225000
     1              12X,'DISTANCE',8X,'CL LN',2X,'CORDINATES')                  00226000
29    FORMAT(' ',I4,2X,5(I4,1X),2X,5(F3.1,1X),5(1X,F3.0),2X,I5,2X,              00227000
     1I5,'/',I5)                                                                00228000
30    FORMAT(12(2X,I8))                                                         00229000
31    FORMAT('1',5X,'THIS SIMULATION RUN WAS MADE ON A NETWORK WITH THE 00230000
     1 FOLLOWING PROPERTIES',//17X,'NUMBER OF NODES = ',I4//17X,'MINIMUM00231000
     2 NUMBER OF ADJACENT NODES = ',I1//17X,'MAXIMUM NUMBER OF ADJACENT 00232000
     3NODES = ',I1//17X,'BUFFER SIZE = ',I5//17X,'LOOKAHEAD LEVEL =',I2/00233000
     4/17X,'BIAS APPLIED TO DISTANCE = ',F5.4//17X,'UPDATE INTERVAL = ',00234000
     5F5.2//17X,'AVERAGE MESSAGE LENGTH = ',F5.1//17X,'MAXIMUM MESSAGE L00235000
     6EVEL = ',I5//17X,'MESSAGE SIMULATION LIMIT = ',I5//17X,'INITIAL EF00236000
     7FECTIVE DATA RATE = ',F5.3//17X,'DATA RATE INCREMENT FOR SUBSEQUEN00237000
     8T RUNS = ',F5.3)                                                          00238000
32    FORMAT(6(2X,F8.3))                                                        00239000
33    FORMAT('1UNABLE TO LOCATE CORRESPONDING ENTRY IN THE ADJACENCY MAT00240000
```

```
IRIX FOR I = ',I4,' AND J = ',I4,' FOR:',//20X,'ADJMAT(I,J) = ',    00241000
25(I4,2X))                                                          00242000
 END                                                                00243000
```

AD-A059 849 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
OPTIMAL ROUTING WITHIN LARGE SCALE DISTRIBUTED COMPUTER-COMMUNI--ETC(U)
MAY 78 W H GREEN

UNCLASSIFIED AFIT-CI-78-69 NL

3 OF 3
AD
A059849



END
DATE
FILMED
12-78
DDC

```
      SUBROUTINE RANDOM(Z)                                                  00245000
C                                                                           00246000
C*************************************************************************00247000
C                                                                           00248000
C     SUBROUTINE FOR GENERATING RANDOM NUMBERS HAVING A UNIFORM             00249000
C     DISTRIBUTION, BY THE MIXED MULTIPLICATIVE CONGRUENTIAL METHOD.        00250000
C                                                                           00251000
C*************************************************************************00252000
C                                                                           00253000
      INTEGER A,X                                                           00254000
      DATA I/1/                                                             00255000
      IF(I.EQ.0)GO TO 1                                                     00256000
          I = 0                                                             00257000
          M = 2**20                                                         00258000
          FM = M                                                            00259000
          X = 566387                                                        00260000
          A = 2**10+3                                                       00261000
C                                                                           00262000
1     X = MOD(A*X,M)                                                        00263000
      FX = X                                                                00264000
      Z = FX/FM                                                             00265000
      RETURN                                                                00266000
      END                                                                   00267000
```

```
      SUBROUTINE INTLC                                                      00269000
C                                                                           00270000
C*************************************************************************00271000
C                                                                           00272000
C     INTLC IS CALLED BY GASP.  IT INITIALIZES THE GASP RANDOM NUMBER       00273000
C     GENERATOR, ZERO'S BUSY ARRAYS AND ESTABLISHES THE FIRST MESSAGE       00274000
C     GENERATION AND DELAY UPDATE FOR EACH NODE.  IT IS ALSO                00275000
C     USED TO RE-INITIALIZE PARAMETERS FOR THE SECOND AND                   00276000
C     SUBSEQUENT RUNS.                                                      00277000
C                                                                           00278000
C*************************************************************************00279000
C                                                                           00280000
      DIMENSION LSED(6)                                                     00281000
      INTEGER ADJMAT,XCORD,YCORD                                            00282000
      REAL ITB,NBUF                                                         00283000
C                                                                           00284000
C     GASP VARIABLES                                                        00285000
C                                                                           00286000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,         00287000
     1NCRDR,NNAPQ,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),       00288000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                00289000
      COMMON /GCOM5/JUNK(24),SSEED(6)                                       00290000
C                                                                           00291000
C     USER VARIABLES                                                        00292000
C                                                                           00293000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1000294000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,       00295000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,00296000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,50C297000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,50C298000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,       00299000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,00300000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY            00301000
      COMMON /UCOM2/REINC,RELIM,FIRST,RESAVE,UDTLIM,NDIGIT                  00302000
      COMMON /UCOM4/IOUT                                                    00303000
      DATA LSED/2135621899,1956987325,2060189405,1980776232,2095230059,     00304000
     111976924637/                                                          00305000
C                                                                           00306000
C     RESET GASP RANDOM NUMBER GENERATOR                                    00307000
      DO 1 L=1,6                                                            00308000
          SSEED(L) = LSED(L)                                                00309000
          IX = -L                                                           00310000
          RNUM = DRAND(IX)                                                  00311000
1     CONTINUE                                                              00312000
C     TEST FOR A SECOND RUN SEQUENCE.                                       00313000
      IF(RE.GT.0.8)GO TO 2                                                  00314000
C     EITHER OF THESE CONDITIONS MEAN  THAT CURRENT RUN                     00315000
C     SEQUENCE IS INCOMPLETE OR ONLY ONE SEQUENCE IS DESIRED.               00316000
```

```
      IF((FIRST.EQ.0.0).OR.(REINC.GT.0.0))GO TO 3                          00317000
1     READ(5,1015)RE,REINC,MAXSPL                                          00318000
C     RE-INITIALIZE FOR SECOND RUN SEQUENCE AND READ IN NEW                00319000
C     SET OF PARAMETERS.                                                   00320000
      FIRST = 0.0                                                          00321000
3     CONTINUE                                                             00322000
C     BUILD MINIMUM DELAY MATRIX, INITIALIZE BUSY LINE, BUSY               00323000
C     LINK, BUSY CPU FLAGS, AND INITIALIZE INPUT, OUTPUT, AND              00324000
C     AND TOTAL QUEUE COUNTERS.                                            00325000
      DO 5 I=1,NUMNOD                                                      00326000
          JMAX = ADJMAT(I,6)                                               00327000
          DO 4 J=1,JMAX                                                    00328000
              DELMAT(I,J) = DISMAT(I,J)                                    00329000
              BSYLIN(I,J) = 0.0                                            00330000
              IQUE(I,J) = 0                                                00331000
4         CONTINUE                                                         00332000
          JQUE(I) = 0                                                      00333000
          CPUBSY(I) = 0.0                                                  00334000
          BSYLNK(I) = 0.0                                                  00335000
          ITB(I) = 0.0                                                     00336000
5     CONTINUE                                                             00337000
C     CALCULATE THE MEAN INTERARRIVAL TIME FOR EACH NODE.                  00338000
      DO 7 I=1,NUMNOD                                                      00339000
          TSUM = 0.0                                                       00340000
          JLIM = ADJMAT(I,6)                                               00341000
          DO 6 J=1,JLIM                                                    00342000
              TSUM = TSUM + TRFMAT(I,J)                                    00343000
6         CONTINUE                                                         00344000
          XMU(I) = XLN/TSUM*0.001                                          00345000
7     CONTINUE                                                             00346000
C     INCREMENT LOAD FACTOR FOR EACH RUN.                                  00347000
      RE = RE+REINC*FIRST                                                  00348000
C     TEST FOR PRINT.                                                      00349000
      IF(NOPRT)10,10,8                                                     00350000
8     DO 9 I=1,NUMNOD                                                      00351000
          WRITE(NPRNT,14)I,(ADJMAT(I,J),J=1,5),(DELMAT(I,K),K=1,5),        00352000
     1    XMU(I)                                                           00353000
9     CONTINUE                                                             00354000
C     INITIALIZE COUNTERS FOR MESSAGES GENERATED AND DELIVERED.            00355000
C     LINE BUSY, NO BUFFER, AND BUSY CPU.                                  00356000
10    MSGNO = 0                                                            00357000
      IOUT = 0                                                             00358000
      LINBSY = 0                                                           00359000
      NOBUFR = 0                                                           00360000
      ICPBSY = 0                                                           00361000
      FIRST = 1.0                                                          00362000
C     SET FIRST MESSAGE AND FIRST DELAY UPDATE TIME FOR ALL NODES.         00363000
      DO 11 I=1,NUMNOD                                                     00364000
```

```
      XMU(I) = XMU(I)/RE                                                00365000
      CALL RANDOM(Z)                                                    00366000
C     MESSAGE GENERATION HAS EXPONENTIAL INTERARRIVAL TIMES             00367000
C     FOR POISSON DISTRIBUTION.                                         00368000
      ATRIB(1) = -XMU(I)*ALOG(Z)                                        00369000
      ATRIB(2) = I+1.                                                   00370000
      CALL FILEM(1)                                                     00371000
      CALL COLCT(ATRIB(1),1)                                            00372000
      GMAT(I) = ATRIB(1)                                                00373000
C     IROUT IS A MEANS OF TURNING OFF THE GENERATION OF MDVS.           00374000
      IF(IROUT.EQ.0)GO TO 11                                            00375000
      ATRIB(1) = UDTINT                                                 00376000
      ATRIB(2) = I+NODQUE                                               00377000
      CALL FILEM(1)                                                     00378000
11    CONTINUE                                                          00379000
      WRITE(NPRNT,12)RE,UDTINT,LKAHD                                    00380000
      RETURN                                                            00381000
12    FORMAT('0THIS RUN USES THE FOLLOWING PARAMETERS:',//10X,          00382000
     1    'BIAS = ',F4.2,//10X,'UPDATE INTERVAL = ',F5.2,               00383000
     2    //10X,'SEARCH DEPTH = ',I2)                                   00384000
13    FORMAT(2(F5.2,2X),I5)                                             00385000
14    FORMAT(' ',2X,I4,2X,5(I4,1X),2X,5(F7.5,1X),2X,F6.4)               00386000
      END                                                               00387000
```

```
      SUBROUTINE EVNTS(IX)                                                    00389000
C                                                                             00390000
C*************************************************************************** 00391000
C                                                                             00392000
C     ARGUMENT IX IS OBTAINED FROM ATRIB(2) BY GASP AND HANDED TO EVNTS       00393000
C     WHICH USES THIS VALUE TO DETERMINE THE TYPE EVENT TAKING PLACE.         00394000
C     HAVING DETERMINED THE TYPE EVENT, IX IS CONVERTED TO THE RESPEC-        00395000
C     TIVE NODE BEING EVALUATED AND HANDED TO THE CALLED ROUTINE.             00396000
C                                                                             00397000
C*************************************************************************** 00398000
C                                                                             00399000
      INTEGER ADJMAT,XCORD,YCORD                                              00400000
      REAL ITB,NBUF                                                           00401000
C                                                                             00402000
C                                                                             00403000
C     GASP VARIABLES                                                          00404000
C                                                                             00405000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,           00406000
     1NCRDR,NNAPC,NNAPT,NNATR,NNFIL,NNO(5125),NNTRY,NPRNT,PPARM(50,4),         00407000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                  00408000
C                                                                             00409000
C     USER VARIBLES                                                           00410000
C                                                                             00411000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(100     00412000
     1124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,        00413000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,      00414000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,50     00415000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,500   00416000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,          00417000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,      00418000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY              00419000
C                                                                             00420000
      IF(IX.LT.1)GO TO 12                                                     00423000
      IF(IX.NE.1)GO TO 1                                                      00424000
      CALL ERROR(1)                                                           00425000
      RETURN                                                                  00426000
C     MESSAGE GENERATION EVENT.                                               00427000
1     IF(IX.GT.MSGEVT)GO TO 2                                                 00428000
      I = IX-1                                                                00429000
      CALL MSGGEN(I)                                                          00430000
      RETURN                                                                  00431000
C     HOST TO IMP MESSAGE ARRIVAL.                                            00432000
2     IF(IX.GT.JSTIMP)GO TO 3                                                 00433000
      I = IX-MSGEVT                                                           00434000
      CALL HSTARV(I)                                                          00435000
      RETURN                                                                  00436000
C     ILLEGAL CODES                                                           00437000
```

```
3       IF(IX.GT.IMPEVT)GO TO 4                                  00438000
        CALL ERROR(2)                                            00439000
        RETURN                                                   00440000
C       IMP PROCESSING ROUTINE                                   00441000
4       IF(IX.GT.NODQUE)GO TO 5                                  00442000
        I = IX-IMPEVT                                            00443000
        CALL CPUSVC(I)                                           00444000
        RETURN                                                   00445000
C       GENERATE DELAY VECTORS.                                  00446000
5       IF(IX.GT.LEVENT)GO TO 6                                  00447000
        I = IX-NODQUE                                            00448000
        CALL GENMDV(I)                                           00449000
        RETURN                                                   00450000
C       RETRANSMIT ON RFNM TIME OUT.                             00451000
6       IF(IX.GT.NFNMTO)GO TO 7                                  00452000
        CALL ERROR(6)                                            00453000
        RETURN                                                   00454000
C       NODE RETRANSMISSION ON ACK TIME OUT.                     00455000
7       IF(IX.GT.NCKTO)GO TO 8                                   00456000
        I = IX-NFNMTO                                            00457000
        CALL NODRET(I)                                           00458000
        RETURN                                                   00459000
8       IF(IX.GT.KMITVT)GO TO 9                                  00460000
        CALL ERROR(3)                                            00461000
        RETURN                                                   00462000
C       LINE TRANSMISSION EVENT.                                 00463000
9       IF(IX.GT.KMITVT+NUMLIN)GO TO 10                          00464000
        I = IX-KMITVT                                            00465000
        CALL LINDEL(I)                                           00466000
        RETURN                                                   00467000
10      IF(IX.GT.INCDVT+NUMLIN)GO TO 11                          00468000
        I = IX-INCDVT                                            00469000
        CALL ENCODE(I)                                           00470000
        RETURN                                                   00471000
11      CALL ERROR(4)                                            00472000
12      CALL ERROR(5)                                            00473000
        RETURN                                                   00474000
C                                                                00475000
        END                                                      00476000
```

```
      SUBROUTINE MSGGEN(I)                                                  00478000
C                                                                           00479000
C*************************************************************************00480000
C                                                                           00481000
C     THIS IS THE MESSAGE GENERATION SUBROUTINE.  EACH TIME                 00482000
C     A MESSAGE EVENT OCCURS, BASED ON THE TRAFFIC DENSITY, A               00483000
C     MESSAGE IS SUBSEQUENTLY GENERATED AND ANOTHER MESSAGE                 00484000
C     EVENT ESTABLISHED FOR THAT NODE.  I IS THE NODE SUPPORT-              00485000
C     ING THE HOST GENERATING THE MESSAGE.                                  00486000
C                                                                           00487000
C*************************************************************************00488000
C                                                                           00489000
      INTEGER ADJMAT,XCORD,YCORD                                            00490000
      REAL ITB,NBUF                                                         00491000
C                                                                           00492000
C     GASP VARIABLES                                                        00493000
C                                                                           00494000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,         00495000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),       00496000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                00497000
C                                                                           00498000
C     USER VARIABLES                                                        00499000
C                                                                           00500000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1000501000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,       00502000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,    00503000
     3BSYLIN(1024,5),PATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,500504000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,500505000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,       00506000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,   00507000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY           00508000
      COMMON /UCOM2/ REINC,PELIM,FIRST,RESAVE,UDTLIM,NDIGIT                 00509000
      COMMON /UCOM4/ IOUT                                                   00510000
C                                                                           00511000
C     SCHEDULE NEXT MESSAGE GENERATION                                      00512000
      CALL RANDOM(Z)                                                        00513000
      ATRIB(1) = TNOW-XMU(I)*ALOG(Z)                                        00514000
      ATRIB(2) = I+1.0                                                      00515000
C     FILE THE EVENT.                                                       00516000
      CALL FILEM (1)                                                        00517000
      GTIME = ATRIB(1)-GMAT(I)                                              00518000
C     ACCUMULATE STATISTICS                                                 00519000
      CALL COLCT(GTIME,1)                                                   00520000
      GMAT(I) = ATRIB(1)                                                    00521000
C     GENERATE SOURCE ADDRESS                                               00522000
      ATRIB(1) = I                                                          00523000
C     GENERATE DESTINATION ADDRESS                                          00524000
1     CALL RANDOM(Z)                                                        00525000
```

```
      IDEST = (NUMNOD-1)*Z+1.0                                          00526000
      IF(IDEST.EQ.I)GO TO 1                                             00527000
      ATRIB(2) = IDEST                                                  00528000
C     GENERATE MESSAGE NUMBER                                           00529000
      MSGNO = MSGNO +1                                                  00530000
      ATRIB(3) = MSGNO                                                  00531000
C     GENERATE MESSAGE LENGTH                                           00532000
      ATRIB(4) = XLN                                                    00533000
C     IDENTIFY MESSAGE TYPE.                                            00534000
      ATRIB(6) = 1.0                                                    00535000
C     GENERATE ORIGINATION TIME                                         00536000
      ATRIB(9) = TNOW                                                   00537000
C     INITIALIZE HOP COUNT                                              00538000
      ATRIB(10) = 0.0                                                   00539000
C     PLACE MESSAGE ON HOST QUEUE                                       00540000
      CALL FILEM(I+1)                                                   00541000
C     COUNT FOR GENERATING INTERIUM OUTPUT.  WHEN COUNT IS              00542000
C     SATISFIED GENERATE OUTPUT AND INITIALIZE COUNTER FOR              00543000
C     NEXT OUTPUT SEQUENCE.                                             00544000
      NCNT = NCNT+1                                                     00545000
      IF(NCNT.LT.ITEM)GO TO 2                                           00546000
      WRITE(NPRNT,3)TNOW,MSGNO,ICUT                                     00547000
      CALL COLCT(0.0,0)                                                 00548000
      NCNT = 0                                                          00549000
C     CHECK FOR FREE LINK AND INPUT BUFFER                              00550000
2     IF(BSYLNK(I).NE.0.0)RETURN                                        00551000
      IF(ITB(I).GE.NBUF/2.0)RETURN                                      00552000
C     RESOURCES ARE AVAILABLE SO GENERATE MESSAGE TRANSMISSION          00553000
C     EVENT.                                                            00554000
      ATRIB(1) = TNOW+XYLN                                              00555000
      ATRIB(2) = I+MSGEVT                                               00556000
      CALL FILEM(I)                                                     00557000
      BSYLNK(I) = TNOW                                                  00558000
      RETURN                                                            00559000
3     FORMAT('0TNOW = ',F12.4,'; MESSAGES GENERATED = ',I7,             00560000
     1'; MESSAGES DELIVERED = ',I6)                                     00561000
      END                                                               00562000
```

```
      SUBROUTINE GENMDV(IX)                                                 00564000
C                                                                           00565000
C*************************************************************************00566000
C                                                                           00567000
C     GENMDV GENERATES A DELAY MESSAGE FOR NODE IDENTIFIED IN IX.  IF       00568000
C     THE QUEUE FOR A PORT IS EMPTY, THE MINIMUM DELAY MESSAGE IS           00569000
C     IMMEDIATELY TRANSMITTED. OTHERWISE, THE MESSAGE IS QUEUED FOR         00570000
C     TRANSMISSION AT THE HIGHEST PRIORITY.  DELAY MESSAGES ARE TRANS-      00571000
C     MITTED TO ALL NODES IDENTIFIED IN AN N-DEPTH SEARCH.                  00572000
C                                                                           00573000
C*************************************************************************00574000
C                                                                           00575000
      INTEGER ADJMAT,XCORD,YCORD                                            00576000
      REAL ITB,NBUF                                                         00577000
C                                                                           00578000
C     GASP VARIABLES                                                        00579000
C                                                                           00580000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,         00581000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),       00582000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                00583000
C                                                                           00584000
C     USER VARIABLES                                                        00585000
C                                                                           00586000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1000587000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,       00588000
     2ITRACE,XMOV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAMD,Z,00589000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,500590000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,500591000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCOVT,BIAS,       00592000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,00593000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY            00594000
C                                                                           00595000
C     SCHEDULE NEXT MDV GENERATION IF NOT TURNED OFF.                       00596000
      IF(IROUT.EQ.0)RETURN                                                  00597000
      A1 = ATRIB(1)                                                         00598000
      ATRIB(1) = TNOW+UDTINT                                                00599000
      CALL FILEM(1)                                                         00600000
      ATRIB(1) = A1                                                         00601000
C     PRESERVE MDV PACKET LENGTH                                            00602000
      ATRIB(4) = 480.                                                       00603000
C     IDENTIFY MESSAGE TYPE                                                 00604000
      ATRIB(6) = 3.0                                                        00605000
C     GENERATE ORIGINATION TIME                                             00606000
      ATRIB(9) = TNOW                                                       00607000
      JMAX = ADJMAT(IX,6)                                                   00608000
C     NODAL DELAY = DISTANCE*8 MICROSECS/MILE + 70MICROSECS (MODEM          00609000
C     DELAY) + 1 MILSECS*NUMBER OF MESSAGES IN RECEIVE QUEUE.               00610000
      DO 1 J=1,JMAX                                                         00611000
```

```
            ATRIB(2) = ADJMAT(IX,J)                                          00612000
            ATRIB(5) = ADJMAT(IX,7)+J                                        00613000
            ATRIB(7) = DISMAT(IX,J)+0.001*ITB(IX)                            00614000
C           ADD IN ONE ADDITIONAL CPU PROCESS TIME IF THE MESSAGE            00615000
C           MUST CYCLE FOR BUSY LINE.                                        00616000
            IF(BSYLIN(IX,J).NE.0.0)ATRIB(7) = ATRIB(7)+0.001                 00617000
C           CPUSV2 SERVICES THE MESSAGE FOR NODE IX.                         00618000
            CALL CPUSV2(IX)                                                  00619000
1       CONTINUE                                                             00620000
        RETURN                                                               00621000
        END                                                                  00622000
```

```
      SUBROUTINE ENCODE (I)                                                    00624000
C                                                                              00625000
C*****************************************************************************00626000
C                                                                              00627000
C     ENCODE SIMULATES LINE TRANSMISSION.  CALLING ARGUMENT IS ASSIGNED        00628000
C     LINK NUMBER.                                                             00629000
C                                                                              00630000
C*****************************************************************************00631000
C                                                                              00632000
      INTEGER ADJMAT,XCORD,YCORD                                               00633000
      REAL ITB,NBUF                                                            00634000
C                                                                              00635000
C     GASP VARIABLES                                                           00636000
C                                                                              00637000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,            00638000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),          00639000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                   00640000
C                                                                              00641000
C     USER VARIABLES                                                           00642000
C                                                                              00643000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(100      00644000
     1124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,         00645000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,00646000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,500647000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,500648000
     5),MSGEVT,JSTIMP,IMPEVT,NFNXTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,          00649000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,00650000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY               00651000
C                                                                              00652000
C     LOCATE SOURCE AND DESTINATION NODES                                      00653000
      IS = LSORC(I)                                                            00654000
      ID = LDEST(I)                                                            00655000
C     FIND THE J POINTER                                                       00656000
      JP = LKPT(I,1)                                                           00657000
C     CHECK FOR LINE BUFFER EMPTY                                              00658000
      IF(NNQ(IS+IMPEVT).NE.0)GO TO 1                                           00659000
      BSYLIN(IS,JP) = 0.0                                                      00660000
      RETURN                                                                   00661000
C     FIND THE MESSAGE REQUIRING SERVICE.                                      00662000
1     SRCH = I                                                                 00663000
      IFND = NFIND(SRCH,5,IS+IMPEVT,5,0.0)                                     00664000
      IF(IFND.NE.0)GO TO 2                                                     00665000
      CALL ERROR(6)                                                            00666000
C     PLACE THE MESSAGE IN THE ATTRIBUTE VECTOR.                               00667000
2     CALL RMOVE(IFND,IS+IMPEVT)                                               00668000
C     FILE THE MESSAGE IN THE MESSAGE EVENT QUEUE.                             00669000
      CALL FILEM(IS+MSGEVT)                                                    00670000
C     TEST FOR RESOURCES.                                                      00671000
```

```
      IF(CPUBSY(IS).NE.0)GO TO 3                                        00672000
      IF(NNQ(IS+JSTIMP).EQ.0)GO TO 3                                    00673000
      IF(ITB(IS).GE.NBUF)GO TO 3                                        00674000
C     RESOURCES ARE AVAILABLE SO GENERATE A NODE EVENT.                 00675000
      CPUBSY(IS) = TNOW                                                 00676000
      ATRIB(1) = TNOW+0.001                                             00677000
      ATRIB(2) = IS+IMPEVT                                              00678000
      CALL FILEM(1)                                                     00679000
C     SCHEDULE NEXT NODE ARRIVAL                                        00680000
3     ATRIB(1) = TNOW+DISMAT(IS,JP)                                     00681000
      ATRIB(2) = I+KMITVT                                               00682000
      CALL FILEM(1)                                                     00683000
C     COLLECT LINE BUSY TIME AND SCHEDULE NEXT ARRIVAL                  00684000
      LINBSY = LINBSY+1                                                 00685000
      BSYLIN(IS,JP) = TNOW                                              00686000
      SRCH = 1                                                          00687000
      IFND = NFIND(SRCH,5,IS+IMPEVT,5,0.0)                              00688000
      IF(IFND.NE.0)GO TO 4                                              00689000
C     NOTHING IN BUFFER - CLEAR BUSY FLAG                               00690000
      BSYLIN(IS,JP) = 0.0                                               00691000
      RETURN                                                            00692000
C     SCHEDULE NEXT MESSAGE ARRIVAL                                     00693000
4     CALL COPY(IFND,IS+IMPEVT)                                         00694000
      ATRIB(1) = TNOW+XZLN                                              00695000
      ATRIB(2) = I+INCDVT                                               00696000
      CALL FILEM(1)                                                     00697000
      RETURN                                                            00698000
      END                                                               00699000
```

```
      SUBROUTINE NODRET(I)                                                    00702000
C                                                                             00703000
C*************************************************************************** 00704000
C                                                                             00705000
C     NODRET IS USED TO RETRANSMIT MESSAGES FROM THE SENT QUEUE UPON          00706000
C     TIME OUT.  CALLING ARGUMENT IS THE NODE NUMBER.                         00707000
C                                                                             00708000
C*************************************************************************** 00709000
C                                                                             00710000
      INTEGER ADJMAT,XCORD,YCORD                                              00711000
      REAL ITB,NBUF                                                           00712000
C                                                                             00713000
C     GASP VARIABLES                                                          00714000
C                                                                             00715000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,           00716000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),         00717000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                  00718000
C                                                                             00719000
C     USER VARIABLES                                                          00720000
C                                                                             00721000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1000722000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,    00723000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,00724000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,500725000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,50C726000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,     00727000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,00728000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY          00729000
C                                                                             00730000
C     CHECK FOR MESSAGE SCRUBBED                                              00731000
      SRCH = ATRIB(3)                                                         00732000
      IFND = NFIND(SRCH,5,I+NODQUE,3,0.0)                                     00733000
      IF(IFND.EQ.0)RETURN                                                     00734000
C     DELETE MESSAGE FROM SENT QUEUE                                          00735000
      CALL RMOVE(IFND,I+NODQUE)                                               00736000
      ID = ATRIB(8)                                                           00737000
C     CHECK FOR INPUT BUFFER SPACE                                            00738000
      IF(ITB(ID).LT.NBUF)GO TO 1                                              00739000
      NOBUFR = NOBUFR+1                                                       00740000
      CALL FILEM(I+NODQUE)                                                    00741000
C     SCHEDULE RETRANSMISSION                                                 00742000
      A1 = ATRIB(1)                                                           00743000
      A2 = ATRIB(2)                                                           00744000
      ATRIB(1) = TNOW+0.1                                                     00745000
      ATRIB(2) = I+NFNMTO                                                     00746000
      CALL FILEM(1)                                                           00747000
      ATRIB(1) = A1                                                           00748000
      ATRIB(2) = A2                                                           00749000
```

```
      RETURN                                                            00750000
C     CALL CPU ROUTINE TO PLACE MESSAGE ON HOST QUEUE                   00751000
1     CALL CPURET(I)                                                    00752000
      RETURN                                                            00753000
      END                                                               00754000
```

```
      SUBROUTINE HSTARV(I)                                                    00756000
C                                                                             00757000
C*************************************************************************** 00758000
C                                                                             00759000
C     THIS ROUTINE ENQUEUES INCOMMING MESSAGES FROM THE HOST.                 00760000
C     CALLING ARGUMENT IS NODE NUMBER.                                        00761000
C                                                                             00762000
C*************************************************************************** 00763000
C                                                                             00764000
      INTEGER ADJMAT,XCORD,YCORD                                              00765000
      REAL ITB,NBUF                                                           00766000
C                                                                             00767000
C     GASP VARIABLES                                                          00768000
C                                                                             00769000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,           00770000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),        00771000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                  00772000
      COMMON /GCOM6/ EENQ(5125),IINN(5125),KKRNK(5125),MMAXQ(5125),           00773000
     1QQTIM(5125),SSQBV(25,5),SSTPV(25,6),VVNQ(5125)                          00774000
C                                                                             00775000
C     USER VARIABLES                                                          00776000
C                                                                             00777000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1000778000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,     00779000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,  00780000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,5 00781000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,5 00782000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,     00783000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM, 00784000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY          00785000
C                                                                             00786000
C     CHECK FOR AVAILABLE SPACE ON INPUT QUEUE                                00787000
      IF(ITB(I).GE.NBUF/2.0)GO TO 2                                           00788000
C     SEE IF HOST QUEUE ENTRY HAS ABORTED                                     00789000
      IF(NNQ(I+1).EQ.0)GO TO 2                                                00790000
C     PICK UP FIRST MESSAGE FROM HOST QUEUE                                   00791000
      CALL RMOVE(MFE(I+1),I+1)                                                00792000
C     FILE IN NODE INPUT QUEUE AND INCREMENT APPROPRIATE                      00793000
C     QUEUE COUNTS.                                                           00794000
      CALL FILEM(I+JSTIMP)                                                    00795000
      JQUE(I) = JQUE(I)+1                                                     00796000
      ITB(I) = ITB(I)+1.0                                                     00797000
C     CHECK FOR BUSY CPU                                                      00798000
      IF(CPUBSY(I).NE.0.0)GO TO 1                                             00799000
C     START PROCESSING ROUTINE AND SCHEDULE END OF PROCESSING                 00800000
      CPUBSY(I) = TNOW                                                        00801000
      ATRIB(1) = TNOW+0.001                                                   00802000
      ATRIB(2) = I+IMPEVT                                                     00803000
```

```
      CALL FILEM(1)                                                    00804000
C     COLLECT STATISTICS ON BUSY LINE TIMES.                           00805000
1     SYSTLK = TNOW-BSYLNK(I)                                          00806000
      CALL COLCT(SYSTLK,2)                                             00807000
      BSYLNK(I) = 0.0                                                  00808000
C     TEST FOR RESOURCES.                                              00809000
      IF(NNQ(I+1).EQ.0)RETURN                                          00810000
      IF(ITB(I).GE.NBUF/2.0)RETURN                                     00811000
C     RESOURCES ARE AVAILABLE SO GENERATE EVENT TO TRANSMIT            00812000
C     NEXT MESSAGE FROM HOST.                                          00813000
      CALL COPY (MFE(I+1),I+1)                                         00814000
      ATRIB(1) = TNOW+XYLN                                             00815000
      ATRIB(2) = I+MSGEVT                                              00816000
      CALL FILEM(1)                                                    00817000
      BSYLNK(I) = TNOW                                                 00818000
      RETURN                                                           00819000
2     BSYLNK(I) = 0.0                                                  00820000
      RETURN                                                           00821000
      END                                                              00822000
```

```
      SUBROUTINE LINDEL(I)                                                    00824000
C                                                                             00825000
C*************************************************************************** 00826000
C                                                                             00827000
C     SIMULATE LINE TRANSMISSION. LINE ERROR AND INITIAL MESSAGE              00828000
C     PROCESSING AND ENQUEUEING.  LINDEL IS CALLED BY EVNTS AND CALLS         00829000
C     UPON SUBROUTINES RFNM, MSG, AND MOV DEPENDING UPON THE TYPE OF          00830000
C     MESSAGE BEING PROCESSED.                                                00831000
C                                                                             00832000
C*************************************************************************** 00833000
C                                                                             00834000
      INTEGER ADJMAT,XCORD,YCORD                                              00835000
      REAL ITB,NBUF                                                           00836000
C                                                                             00837000
C     GASP VARIABLES                                                          00838000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,           00839000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),         00840000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                  00841000
C                                                                             00842000
C     USER VARIABLES                                                          00843000
C                                                                             00844000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1000845000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,     00846000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,00847000
     38SYLIN(1024,5),MATMAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,500848000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,500849000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,     00850000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,00851000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY          00852000
C                                                                             00853000
C     FIND SOURCE AND DESTINATION NODES                                       00854000
      IS = LSORC(I)                                                           00855000
      ID = LDEST(I)                                                           00856000
C     FIND RETURNING LINE NUMBER                                              00857000
      LINRET = LKPT(I,2)                                                      00858000
C     FIND THE J POINTER                                                      00859000
      J = LKPT(LINRET,1)                                                      00860000
      JP = LKPT(I,1)                                                          00861000
C     CHECK FOR LINE BUFFER EMPTY                                             00862000
      IF(NNQ(MSGEVT+IS).EQ.0)RETURN                                           00863000
C     LOCATE THE MESSAGE NEEDING SERVICE.                                     00864000
      SRCH = I                                                                00865000
      IFND = NFIND(SRCH,5,MSGEVT+IS,5,0,0)                                    00866000
      IF(IFND.NE.0)GO TO 1                                                    00867000
      CALL ERROR(6)                                                           00868000
C     REMOVE MSG FROM NODE LINK QUEUE AND TOTAL QUEUE COUNT.                  00869000
1     IQUE(IS,JP) = IQUE(IS,JP)-1                                             00870000
      ITB(IS) = ITB(IS)-1.0                                                   00871000
```

```
      CALL RMOVE(IFND,IS+MSGEVT)                                        00872000
C     RETRIEVE THE MESSAGE TYPE.                                        00873000
      MSGTYP = ATRIB(6)                                                 00874000
C     CHECK FOR INPUT BUFFER SPACE                                      00875000
      IF(ITB(ID).LT.NBUF)GO TO 2                                        00876000
C     COUNT THE OCCURRENCE OF NO BUFFER AND PLACE THE MESSAGE           00877000
C     IN CURRENT SOURCE QUEUE.                                          00878000
      NOBUFR = NOBUFR+1                                                 00879000
      CALL FILEM(IS+NODQUE)                                             00880000
C     SCHEDULE RETRANSMISSION                                           00881000
      A1 = ATRIB(1)                                                     00882000
      A2 = ATRIB(2)                                                     00883000
      ATRIB(1) = TNOW+0.1                                               00884000
      ATRIB(2) = IS+NFNMTO                                              00885000
      CALL FILEM(1)                                                     00886000
      ATRIB(1) = A1                                                     00887000
      ATRIB(2) = A2                                                     00888000
      GO TO 6                                                           00889000
C     PASS CONTROL TO MESSAGE HANDLING ROUTINES                         00890000
2     GO TO (3,3,4,6),MSGTYP                                            00891000
C     ILLEGAL MESSAGE TYPE = ABORT                                      00892000
      CALL ERROR(6)                                                     00893000
3     CALL MSG(I,IS,ID,LINRET,J)                                        00894000
      GO TO 5                                                           00895000
C     ITS A MDV SO UPDATE DELAY MATRIX.                                 00896000
4     DELMAT(ID,J) = ATRIB(7)                                           00897000
      GO TO 6                                                           00898000
C     IF CPU IDLE, START SERVICE EVENT                                  00899000
5     IF(CPUBSY(ID).NE.0.0)GO TO 6                                      00900000
      CPUBSY(ID) = TNOW                                                 00901000
      A1 = ATRIB(1)                                                     00902000
      A2 = ATRIB(2)                                                     00903000
      ATRIB(1) = TNOW+0.001                                             00904000
      ATRIB(2) = ID+IMPEVT                                              00905000
      CALL FILEM(1)                                                     00906000
      ATRIB(1) = A1                                                     00907000
      ATRIB(2) = A2                                                     00908000
6     RETURN                                                            00909000
      END                                                               00910000
```

```
      SUBROUTINE MSG(I,IS,ID,L,J)                                          00912000
C                                                                          00913000
C*************************************************************************00914000
C                                                                          00915000
C     THIS SUBROUTINE IS CALLED BY LINDEL WHICH HAS DETECTED A MESSAGE     00916000
C     PROCESSING EVENT.  IF THE MESSAGE HAS REACHED ITS DESTINATION,       00917000
C     STATISTICS ARE COLLECTED AND A RFNM GENERATED (IF NOT SUPPRESSED).   00918000
C     THE MESSAGE IS THEN ACKED.                                           00919000
C                                                                          00920000
C*************************************************************************00921000
C                                                                          00922000
      INTEGER ADJMAT,XCORD,YCORD                                           00923000
      REAL ITB,NBUF                                                        00924000
C                                                                          00925000
C     GASP VARIABLES                                                       00926000
C                                                                          00927000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,        00928000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),      00929000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                               00930000
C                                                                          00931000
C     USER VARIABLES                                                       00932000
C                                                                          00933000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1    00934000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,      00935000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,   00936000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,5   00937000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,5   00938000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,      00939000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,   00940000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY           00941000
      COMMON /UCOM4/ IOUT                                                  00942000
C                                                                          00943000
      ATRIB(10) = ATRIB(10)+1.0                                            00944000
C     INCREMENT HOP COUNT AND HOP INHIBIT COUNTER.                         00945000
      ATRIB(7) = ATRIB(7)+1.0                                              00946000
C     COLLECT DESTINATION TO DETERMINE IF REACHED.                         00947000
      IDN = ATRIB(2)                                                       00948000
      IF(IDN.NE.ID)GO TO 5                                                 00949000
C     DELIVER TO HOST AND COLLECT DELAY STATISTICS                         00950000
      IOUT = IOUT+1                                                        00951000
      IF(IOUT.GT.MAXSPL)MSTOP=-1                                           00952000
C     COLLECT HOP COUNT AND DELAY STATISTICS.                              00953000
      DTIME = TNOW-ATRIB(9)+XYLN                                           00954000
      CALL COLCT(DTIME,4)                                                  00955000
      CALL COLCT(ATRIB(10),3)                                              00956000
C     INCREMENT COUNTER FOR PRINT TEST, PRINT INTERIUM OUTPUT IF COUNTER   00957000
C     REACHES THRESHOLD.                                                   00958000
      NAK = NAK+1                                                          00959000
```

```
      IF(NAK.LT.ITEM)GO TO 10                                          00960000
      WRITE(6,1010)TNOW,MSGNO,IOUT                                     00961000
      CALL OTPUT                                                       00962000
      NAK = 0                                                          00963000
      GO TO 10                                                         00964000
C     PLACE ON NODE QUEUE                                              00965000
5     CALL FILEM(ID+JSTIMP)                                            00966000
C     INCREMENT OUTPUT AND TOTAL  MESSAGE QUEUE COUNTERS.              00967000
      JQUE(ID) = JQUE(ID)+1                                            00968000
      ITB(ID) = ITB(ID)+1.0                                            00969000
C     GENERATE ACK AND QUEUE IN OUTPUT                                 00970000
10    ATRIB(1) = ID                                                    00971000
      ATRIB(2) = IS                                                    00972000
      ATRIB(4) = 118.0                                                 00973000
      ATRIB(5) = L                                                     00974000
      ATRIB(6) = 4.0                                                   00975000
      ATRIB(9) = TNOW                                                  00976000
      MATHAG(KSTOR,2) = J                                              00977000
      CALL CPUSV1(ID)                                                  00978000
      RETURN                                                           00979000
1010  FORMAT('0TNOW = ',F12.4,'; MESSAGES GENERATED = ',I7,            00980000
     1'; MESSAGES DELIVERED = ',I6)                                    00981000
      END                                                              00982000
```

```
      SUBROUTINE CPUSVC(I)                                                  00984000
C                                                                           00985000
C*************************************************************************00986000
C                                                                           00987000
C     CPUSVC IS A MULTI-SERVICE ROUTINE CALLED BY SEVERAL OTHER             00988000
C     SUBROUTINES TO SIMULATE NODE SERVICE TIME FOR ANY TYPE MESSAGE        00989000
C     BEING SERVICED.  THE CALL ARGUMENT IS THE NODE NUMBER.                00990000
C                                                                           00991000
C*************************************************************************00992000
C                                                                           00993000
      INTEGER ADJMAT,XCORD,YCORD                                            00994000
      REAL ITB,NBUF                                                         00995000
C                                                                           00996000
C     GASP VARIABLES                                                        00997000
C                                                                           00998000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,         00999000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),       01000000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                01001000
C                                                                           01002000
C     USER VARIABLES                                                        01003000
C                                                                           01004000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(100 01005000
     1124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,     01006000
     2ITRACE,XMOV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,   01007000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,50 01008000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,50 01009000
     5),MSGEVT,JSTIMP,IMPEVT,NFNRTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,      01010000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,  01011000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY           01012000
      COMMON /UCOM4/IOUT                                                    01013000
      DATA IENT/0/                                                          01014000
C                                                                           01015000
C     CHECK FOR OUTPUT BUFFER                                               01016000
      IENT = 1                                                              01017000
      IF(ITB(I).GE.NBUF)GO TO 6                                             01018000
C     CHECK FOR MESSAGE ON INPUT QUEUE AND REMOVE IF FOUND.                 01019000
      IF(NNQ(I+JSTIMP).LE.0)GO TO 6                                         01020000
      CALL RMOVE(MFE(I+JSTIMP),I+JSTIMP)                                    01021000
C     DECREMENT COUNTERS IF MESSAGE REMOVED.                                01022000
      JQUE(I) = JQUE(I)-1                                                   01023000
      ITB(I) = ITB(I)-1.0                                                   01024000
C     IF AN MOV, ROUTE IS ADJACENT NODE SO GO TO SCHEDULE MSG DELIVERY.     01025000
      IF(ATRIB(6).NE.3.0)GO TO 1                                            01026000
      ENTRY CPUSV2(I)                                                       01027000
C     COLLECT RETURN LINE.                                                  01028000
      LINRET = ATRIB(5)                                                     01029000
C     COLLECT POINTER J TO NEXT NODE IN ADJACENCY MATRIX.                   01030000
      NXNDJ = LKPT(LINRET,1)                                                01031000
```

```
      GO TO 2                                                          01032000
      ENTRY CPURET(I)                                                  01033000
C     CALL ROUTING ALGORITHM.  NEXT NODE IS RETURNED IN ATRIB(8).      01034000
1     CALL RTALGC(I)                                                   01035000
      ENTRY CPUSVI(I)                                                  01036000
C     COLLECT J POINTER TO NEXT NODE.                                  01037000
      NXNDJ = MATHAG(KSTOR,2)                                          01038000
C     COLLECT RETURN LINE AND PLACE IN ATRIB(5).                       01039000
      LINRET = ADJMAT(I,7)+NXNDJ                                       01040000
      ATRIB(5) = LINRET                                                01041000
C     BECOMES AN IMP PROCESSING EVENT                                  01042000
2     CONTINUE                                                         01043000
C     TEST FOR INVALID J POINTER.                                      01044000
      IF((NXNDJ.LE.ADJMAT(I,6)).AND.(NXNDJ.GT.0))GO TO 3               01045000
      WRITE(6,10)NXNDJ,I                                               01046000
      CALL ERROR(2)                                                    01047000
C     FILE MESSAGE IN NODE QUEUE AND INCREMENT QUEUE COUNTERS.         01048000
3     CALL FILEM(I+IMPEVT)                                             01049000
      IQUE(I,NXNDJ) = IQUE(I,NXNDJ)+1                                  01050000
      ITB(I) = ITB(I)+1.0                                              01051000
C     IF LINE NOT BUSY TRANSMIT MESSAGE                                01052000
4     IF(BSYLIN(I,NXNDJ).NE.0.0)GO TO 5                                01053000
C     SCHEDULE MESSAGE DELIVERY                                        01054000
      ATRIB(1) = TNOW+XZLN                                             01055000
      ATRIB(2) = INCOMT+LINRET                                         01056000
      CALL FILEM(1)                                                    01057000
C     FLAG LINE TO NEXT NODE AS BUSY.                                  01058000
      BSYLIN(I,NXNDJ) = TNOW                                           01059000
C     CPU GOING IDLE. COLLECT CPU BUSY STATISTICS                      01060000
5     IF(IENT.EQ.0)GO TO 7                                             01061000
      ICPBSY = ICPBSY+1                                                01062000
6     CPUBSY(I) = 0.0                                                  01063000
C     IF MESSAGES IN INPUT QUEUE, SCHEDULE NEXT SERVICE EVENT.         01064000
7     IF(CPUBSY(I).NE.0.0)GO TO 8                                      01065000
      IF(NNQ(I+JSTIMP).LE.0)GO TO 8                                    01066000
      ATRIB(1) = TNOW+0.001                                            01067000
      ATRIB(2) = I+IMPEVT                                              01068000
      CALL FILEM(1)                                                    01069000
C     FLAG CPU AS BUSY.                                                01070000
      CPUBSY(I) = TNOW                                                 01071000
C     TEST FOR RESOURCES.                                              01072000
8     IF(ITB(I).GE.NBUF/2.0)GO TO 9                                    01073000
      IF(NNQ(I+1).LE.0)GO TO 9                                         01074000
      IF(BSYLNK(I).NE.0.0)GO TO 9                                      01075000
C     GENERATE EVENT IF RESOURCES AVAILABLE.                           01076000
      ATRIB(1) = TNOW+XYLN                                             01077000
      ATRIB(2) = I+MSGEVT                                              01078000
      CALL FILEM(1)                                                    01079000
```

```
      BSYLNK(I) = TNOW                                                  01080000
9     IENT = 0                                                          01081000
C     COLLECT STATISTICS ON ALL INPUT QUEUES.                           01082000
      Q = ITB(I)                                                        01083000
      CALL COLCT(Q,5)                                                   01084000
      RETURN                                                            01085000
10    FORMAT('0THE J POINTER NXNCJ = ',I4,' IN ROUTINE CPUSVC IS OUT OF 01086000
     1LIMITS FOR I = ',I4)                                              01087000
      END                                                               01088000
```

```
      SUBROUTINE RTALCO(I)                                                    01090000
C                                                                             01091000
C*************************************************************************** 01092000
C                                                                             01093000
C     DETERMINES THE ROUTING SCHEME BY EVALUATING WHICH NODE IS CLOSER        01094000
C     TO THE DESTINATION NODE AT AN LKAHD-DEPTH LOOK-AHEAD SEARCH.            01095000
C     CALLING ARGUMENT IS THE NODE NEEDING TO TRANSMIT A MESSAGE              01096000
C                                                                             01097000
C*************************************************************************** 01098000
C                                                                             01099000
      INTEGER ADJMAT,XCORD,YCORD                                              01100000
      REAL ITB,NBUF                                                           01101000
      DIMENSION TRCVEC(4),CATHAG(625,2)                                       01102000
C                                                                             01103000
C     GASP VARIABLES                                                          01104000
C                                                                             01105000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,           01106000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNO(5125),NNTRY,NPRNT,PPARM(50,4),         01107000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                  01108000
C                                                                             01109000
C     USER VARIABLES                                                          01110000
C                                                                             01111000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(100 01112000
     1124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,    01113000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,01114000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,501115000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,501116000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,     01117000
     6NODQUE,NOPRT,IMAT(6),IMATPT,6),IVALS(6),GMAT(1024),ITB(1024),NRFM,01118000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY          01119000
      COMMON /UCOM2/ REINC,RELIM,FIRST,RESAVE,UDTLIM,NDIGIT                01120000
      COMMON /UCOM4/ IOUT                                                  01121000
C                                                                             01122000
C     INITIALIZE ARC POINTERS THROUGH EVALUATION TREE.                        01123000
      J2 = 0                                                                  01124000
      J3 = 0                                                                  01125000
      J4 = 0                                                                  01126000
C     INITIALIZE ARRAY INDICES.                                               01127000
      KSTOR = 1                                                               01128000
      INDEX = 1                                                               01129000
C     SAVE THE SPECIFIED SEARCH DEPTH IN CASE IT IS MODIFIED IN TREE          01130000
C     EVALUATION DUE TO EARLY ENCOUNTER OF DESTINATION.                       01131000
      ISAVE = LKAHD                                                           01132000
C     COLLECT DESTINATION, RETURN LINE, AND HOPS.                             01133000
      IDEST = ATRIB(2)                                                        01134000
      INLIN = ATRIB(5)                                                        01135000
      IHOPS = ATRIB(7)                                                        01136000
C     START THE TREE EVALUATION.  FIRST OBTAIN THE NUMBER OF ADJACENT         01137000
```

```
C     NODES.                                                          01138000
      J1MAX = ADJMAT(I,6)                                             01139000
C     TEST FOR ONLY ONE ADJACENT NODE.                                01140000
      IF(J1MAX.GT.2)GO TO 2                                           01141000
C     IF THIS LINE POINTS TO SENDING NODE, SKIP.                      01142000
      DO 1 J1=1,J1MAX                                                 01143000
          IF(ADJMAT(I,J1).EQ.LSORC(INLIN))GO TO 1                     01144000
C         COLLECT NEXT NODE AND RETURN.                               01145000
          ATRIB(8) = ADJMAT(I,J1)                                     01146000
          MATHAG(1,2) = J1                                            01147000
          GO TO 17                                                    01148000
1     CONTINUE                                                        01149000
C     START SEARCH DEPTH GREATER THAN ONE.                            01150000
2     DO 15 J1=1,J1MAX                                                01151000
C     IF INCOMMING LINE, SKIP.                                        01152000
          IF(ADJMAT(I,J1).EQ.LSORC(INLIN))GO TO 15                    01153000
C         KEEP PATH TRACE.                                            01154000
          TRCVEC(1) = J1                                              01155000
C         GET POINTER FOR NEXT LEVEL IN TREE SEARCH.                  01156000
          I1DEP = ADJMAT(I,J1)                                        01157000
C         COLLECT NEXT NODE AND ADJUST SEARCH DEPTH IF DESTINATION IS 01158000
C         FOUND.                                                      01159000
          ATRIB(8) = I1DEP                                            01160000
          IF(I1DEP.NE.IDEST)GO TO 3                                   01161000
          LKAHD = ISAVE                                               01162000
C         SAVE POINTER FOR LOCATING MIN VALUE.                        01163000
          MATHAG(1,2) = J1                                            01164000
          GO TO 17                                                    01165000
C         IF SEARCH DEPTH ADJUSTED TO 1, EVALUATE COORDINATES.        01166000
3         IF(LKAHD.GT.1)GO TO 4                                       01167000
          MATHAG(INDEX,2) = J1                                        01168000
          CALL PATHAG(I1DEP,INDEX,TRCVEC,I,IHOPS,CATHAG)              01169000
          GO TO 15                                                    01170000
C         COLLECT POINTER FOR NEXT LEVEL OF TREE SEARCH.              01171000
4         J2MAX = ADJMAT(I1DEP,6)                                     01172000
          DO 14 J2=1,J2MAX                                            01173000
C             NOW AT THE SECOND LEVEL. THE ROUTINE IS AS ABOVE.       01174000
              TRCVEC(2) = J2                                          01175000
              I2DEP = ADJMAT(I1DEP,J2)                                01176000
              IF(I2DEP.NE.IDEST)GO TO 5                               01177000
              LKAHD = 2                                               01178000
              GO TO 6                                                 01179000
C             DO NOT EVALUATE NODE JUST EVALUATED.                    01180000
5             IF(I2DEP.EQ.I)GO TO 14                                  01181000
              IF(LKAHD.GT.2)GO TO 7                                   01182000
6             MATHAG(INDEX,2) = J1                                    01183000
              CALL PATHAG(I2DEP,INDEX,TRCVEC,I,IHOPS,CATHAG)          01184000
              GO TO 14                                                01185000
```

```
C               NOW AT THE THIRD LEVEL. RCUTINE REMAINS THE SAME.          01186000
7               J3MAX = ADJMAT(I2DEP,6)                                    01187000
                DO 13 J3=1,J3MAX                                           01188000
                    TRCVEC(3) = J3                                         01189000
                    I3DEP = ADJMAT(I2DEP,J3)                               01190000
                    IF(I3DEP.NE.IDEST)GO TO 8                              01191000
                    LKAHD = 3                                              01192000
                    GO TO 27                                               01193000
C                   DO NOT EVALUATE NODES PREVIOUSLY EVALUATED.            01194000
8                   IF((I3DEP.EQ.I1DEP).OR.(I3DEP.EQ.I))GO TO 13           01195000
                    IF(LKAHD.GT.3)GO TC 10                                 01196000
9                   MATHAG(INDEX,2) = J1                                   01197000
                    CALL PATHAG(I3DEP,INDEX,TRCVEC,I,IHOPS,CATHAG)         01198000
                    GO TO 13                                               01199000
C                   NOW AT THE FORTH AND DEEPEST LEVEL.                    01200000
10                  J4MAX = ADJMAT(I3DEP,6)                                01201000
                    DO 12 J4=1,J4MAX                                       01202000
                        TRCVEC(4) = J4                                     01203000
                        I4DEP = ADJMAT(I3DEP,J4)                           01204000
C                       DO NOT EVALUATE NODES PREVIOUSLY EVALUATED.        01205000
11                      IF((I4DEP.EQ.I2DEP).OR.(I4DEP.EQ.I1DEP).OR.        01206000
     1                      (I4DEP.EQ.I))GO TO 12                          01207000
                        MATHAG(INDEX,2) = J1                               01208000
                        CALL PATHAG (I4DEP,INDEX,TRCVEC,I,IHOPS,CATHAG)    01209000
12                  CONTINUE                                               01210000
13              CONTINUE                                                   01211000
14          CONTINUE                                                       01212000
15      CONTINUE                                                           01213000
C       INITIALIZE FOR FINDING PATH IN TREE OF MIN COST.                   01214000
        KMAX = INDEX-1                                                     01215000
        BIN = CATHAG(1,1)                                                  01216000
        ATRIB(8) = ADJMAT(I,MATHAG(1,2))                                   01217000
C       COMPARE FOR MIN COST.                                              01218000
        DO 16 K=2,KMAX                                                     01219000
            IF(BIN.LE.CATHAG(K,1))GO TO 16                                 01220000
            ATRIB(8) = ADJMAT(I,MATHAG(K,2))                               01221000
            BIN = CATHAG(K,1)                                              01222000
            KSTOR = K                                                      01223000
16      CONTINUE                                                           01224000
C       FIND AND SUPPRESS LOOP MECHANISM.                                  01225000
17      IF(IHOPS.LE.NREM)GO TO 18                                          01226000
        KRIB = IHOPS-NREM                                                  01227000
        IF(KRIB.LE.ISAVE)GO TO 18                                          01228000
        ATRIB(7) = ISAVE                                                   01229000
C       RETURN SEARCH DEPTH VALUE.                                         01230000
18      LKAHD = ISAVE                                                      01231000
        RETURN                                                             01232000
        END                                                                01233000
```

```
      SUBROUTINE PATHAG(NXTNOD,INDEX,TRCVEC,I,IHOPS,CATHAG)                 01235000
C                                                                           01236000
C*************************************************************************01237000
C                                                                           01238000
C     APPLYING THE PATHAGORIAN THEORM, THIS ROUTINE EVALUATES THE DIS-      01239000
C     TANCE BETWEEN NXTNOD AND THE DESTINATION NODE.                        01240000
C                                                                           01241000
C*************************************************************************01242000
C                                                                           01243000
      INTEGER ADJMAT,XCORD,YCORD                                            01244000
      REAL ITB,NBUF                                                         01245000
      DIMENSION TRCVEC(4),CATHAG(625,2)                                     01246000
C                                                                           01247000
C     GASP VARIABLES                                                        01248000
C                                                                           01249000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,         01250000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),      01251000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                01252000
C                                                                           01253000
C                                                                           01254000
C     USER VARIABLES                                                        01255000
C                                                                           01256000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(1001257000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,       01258000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z,    01259000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,501260000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,501261000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,       01262000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM,01263000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY            01264000
      COMMON /UCOM2/ REINC,RELIM,FIRST,RESAVE,UDTLIM,NDIGIT                 01265000
      COMMON /UCOM4/ IOUT                                                   01266000
C                                                                           01267000
C     SAVE THE BIAS FOR POSSIBLE ADJUSTMENT.                                01268000
      SBIAS = BIAS                                                          01269000
C     DO WE IMPLEMENT HOP SUPPRESS MECHANISM?                               01270000
      IF(IHOPS.LE.NRFM)GO TO 1                                              01271000
      BIAS = 0.0                                                            01272000
C     HOP SUPPRESS MECHANISM IMPLEMENTED.  NO NEED TO EVALUATE COORDI-      01273000
C     NATES.  MIN COST WILL BE DETERMINED ON THE TREE LOADS ONLY.           01274000
      GO TO 2                                                               01275000
C     COLLECT DESTINATION AND EVALUATE COORDINATES.                         01276000
1     IDEST = ATRIB(2)                                                      01277000
      ACORD = IABS(XCORD(IDEST)-XCORD(NXTNOD))                              01278000
      BCORD = IABS(YCORD(IDEST)-YCORD(NXTNOD))                              01279000
      CATHAG(INDEX,1) = 0.001*(ACORD+BCORD)+5.0                             01280000
2     TOTDEL = 0.0                                                          01281000
      IDEX = I                                                              01282000
```

```
C     DETERMINE NODAL DELAY IN TREE USING INVERSE EXPONENTIAL BIAS.      01283000
      DO 8 J=1,LKAHD                                                    01284000
          JDEX = TRCVEC(J)                                              01285000
          GO TO (3,4,5,6),J                                             01286000
3         TOTDEL = TOTDEL + DELMAT(IDEX,JDEX)                           01287000
          GO TO 7                                                       01288000
4         TOTDEL = TOTDEL+0.367879*DELMAT(IDEX,JDEX)                    01289000
          GO TO 7                                                       01290000
5         TOTDEL = TOTDEL+0.135335*DELMAT(IDEX,JDEX)                    01291000
          GO TO 7                                                       01292000
6         TOTDEL = TOTDEL+0.049282*DELMAT(IDEX,JDEX)                    01293000
7         IDEX = ADJMAT(IDEX,JDEX)                                      01294000
8     CONTINUE                                                          01295000
C     COLLECT DELAY INTO ARRAY AND INCREMENT POINTER FOR NEXT           01296000
C     EVALUATION.                                                       01297000
      CATHAG(INDEX,1) = CATHAG(INDEX,1)*BIAS+TOTDEL                     01298000
      INDEX = INDEX+1                                                   01299000
C     RETURN BIAS VALUE.                                                01300000
      BIAS = SBIAS                                                      01301000
      RETURN                                                            01302000
      END                                                               01303000
```

```
      SUBROUTINE OTPUT                                                          01305000
C                                                                               01306000
C*****************************************************************************01307000
C                                                                               01308000
C     COLLECT FINAL STATISTICS AND PRINT OUTPUT ALONG WITH PARAMETERS.          01309000
C                                                                               01310000
C*****************************************************************************01311000
C                                                                               01312000
      INTEGER ADJMAT,XCORD,YCORD                                                01313000
      REAL ITB,NBUF                                                             01314000
C                                                                               01315000
C     GASP VARIABLES                                                            01316000
C                                                                               01317000
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(5125),MLE(5125),MSTOP,             01318000
     1NCRDR,NNAPO,NNAPT,NNATR,NNFIL,NNQ(5125),NNTRY,NPRNT,PPARM(50,4),          01319000
     2TNOW,TIBEG,TTCLR,TTFIN,TTRIB(25),TTSET                                    01320000
      COMMON /GCOM6/ EENQ(5125),IINN(5125),KKRNK(5125),MMAXQ(5125),             01321000
     1QQTIM(5125),SSOBV(25,5),SSTPV(25,6),VVNQ(5125)                            01322000
C                                                                               01323000
C     USER VARIABLES                                                            01324000
C                                                                               01325000
      COMMON /UCOM1/ ADJMAT(1025,7),TRFMAT(1024,5),CAPMAT(1024,5),XMU(100 01326000
     124),RE,MSGNO,XLN,IROUT,CPUBSY(1024),BSYLNK(1024),UDTINT,MAXSPL,    01327000
     2ITRACE,XMDV,DP,NBUF,ITEM,NAK,NCNT,XCORD(1024),YCORD(1024),LKAHD,Z, 01328000
     3BSYLIN(1024,5),MATHAG(625,2),LSORC(5120),LDEST(5120),DISMAT(1024,5 01329000
     4),JQUE(1024),IQUE(1024,5),NUMLIN,KSTOR,NUMNOD,MSGLVL,DELMAT(1024,5 01330000
     5),MSGEVT,JSTIMP,IMPEVT,NFNMTO,LEVENT,NCKTO,KMITVT,INCDVT,BIAS,    01331000
     6NODQUE,NOPRT,IMAT(6),IMATPT(6),IVALS(6),GMAT(1024),ITB(1024),NRFM, 01332000
     7LINMAT(1024,5),XYLN,XZLN,LKPT(5120,2),LINBSY,NOBUFR,ICPBSY         01333000
      COMMON /UCOM2/ RELINC,RELIM,FIRST,RESAVE,UDTLIM,NDIGIT                    01334000
C                                                                               01335000
      WRITE(6,4)                                                                01336000
      CALL COLCT(0.0,0)                                                         01337000
      WRITE(6,5)RE                                                              01338000
C     CALCULATE THE OBSERVED SYSTEM LOAD.                                       01339000
      OBSLO = ((MSGNO/TNOW)*(SSOBV(2,1)/SSOBV(2,3))*256000.0)/NDIGIT            01340000
      WRITE(6,6)OBSLO                                                           01341000
      WRITE(6,7)MSGNO                                                           01342000
      WRITE(6,8)TNOW                                                            01343000
      WRITE(6,9)NBUF                                                            01344000
      WRITE(6,10)UDTINT                                                         01345000
      WRITE(6,11)BIAS                                                           01346000
      WRITE(6,14)LINBSY                                                         01347000
      WRITE(6,15)NOBUFR                                                         01348000
      WRITE(6,16)ICPBSY                                                         01349000
      IF(NOPRT)3,3,1                                                            01350000
1     WRITE(6,12)                                                               01351000
      DO 10 I=1,NUMNOD                                                          01352000
```

```
      WRITE(6,13)I,(ADJMAT(I,J),J=1,5),(DELMAT(I,K),K=1,5),          01353000
     1         (IQUE(I,L),L=1,5),JQUE(I),ITB(I)                     01354000
2     CONTINUE                                                       01355000
3     CONTINUE                                                       01356000
      RETURN                                                         01357000
C                                                                    01358000
4     FORMAT('0',59X,'RUN STATISTICS',59X/)                          01359000
5     FORMAT('0',10X,'SYSTEM LOADING FACTOR = ',F10.6)               01360000
6     FORMAT('0',10X,'OBSERVED SYSTEM LOAD = ',F12.6)                01361000
7     FORMAT('0',10X,'LAST MESSAGE GENERATED = ',I10)                01362000
8     FORMAT('0',10X,'CURRENT TIME = ',F12.4)                        01363000
9     FORMAT('0',10X,'NUMBER OF BUFFERS PER NODE = ',F4.0)           01364000
10    FORMAT('0',10X,'UPDATE INTERVAL = ',F10.6)                     01365000
11    FORMAT('0',10X,'LINE BIAS = ',F10.6)                           01366000
12    FORMAT('1',2X,'NODE',7X,'ADJACENT NODES',18X,'DELAY MATRIX',32X,01367000
     1    'SEND QUEUE',10X,'IN QUEUE',1X,'TOT QUEUE')                01368000
13    FORMAT(' ',2X,I4,2X,5(I4,1X),2X,5(F7.5,1X),2X,5(I5,1X),2X,I5,6X,01369000
     1    F4.0)                                                      01370000
14    FORMAT('0',10X,'REJECTIONS FOR LINE BUSY = ',I10)              01371000
15    FORMAT('0',10X,'REJECTIONS FOR BUFFER FULL = ',I10)            01372000
16    FORMAT('0',10X,'REJECTIONS FOR CPU BUSY = ',I10)               01373000
C                                                                    01374000
      END                                                            01375000
```

VITA

William H. Greene was born in Sebring, Florida on August 6, 1938. He entered the U.S. Air Force in June, 1956, immediately after graduating from Madison High School, Madison, Florida. Having been commissioned a Second Lieutenant in 1963, he graduated from Omaha University with a B.G.S. in Mathematics in 1967. Later, in 1972, he graduated from Texas A&M University with a M.C.S. The author has been on continuous active duty in the U.S. Air Force since 1956 and currently holds the rank of Major. As well as having attended Officers' Candidate School, he has completed the Officers' Communications School, Squadron Officers' School, Staff Communications Officer School, and Air Command and Staff School. His forthcoming assignment is as a Staff Communications Officer with Assistant Deputy Chief of Staff for Command and Control located at the Pentagon. Awards and decorations include the Bronze Star Medal and the Meritorious Service Medal. Major Greene is a member of ACM, IEEE, and the UPE honor society.

The typist for this dissertation was Beth Baker.